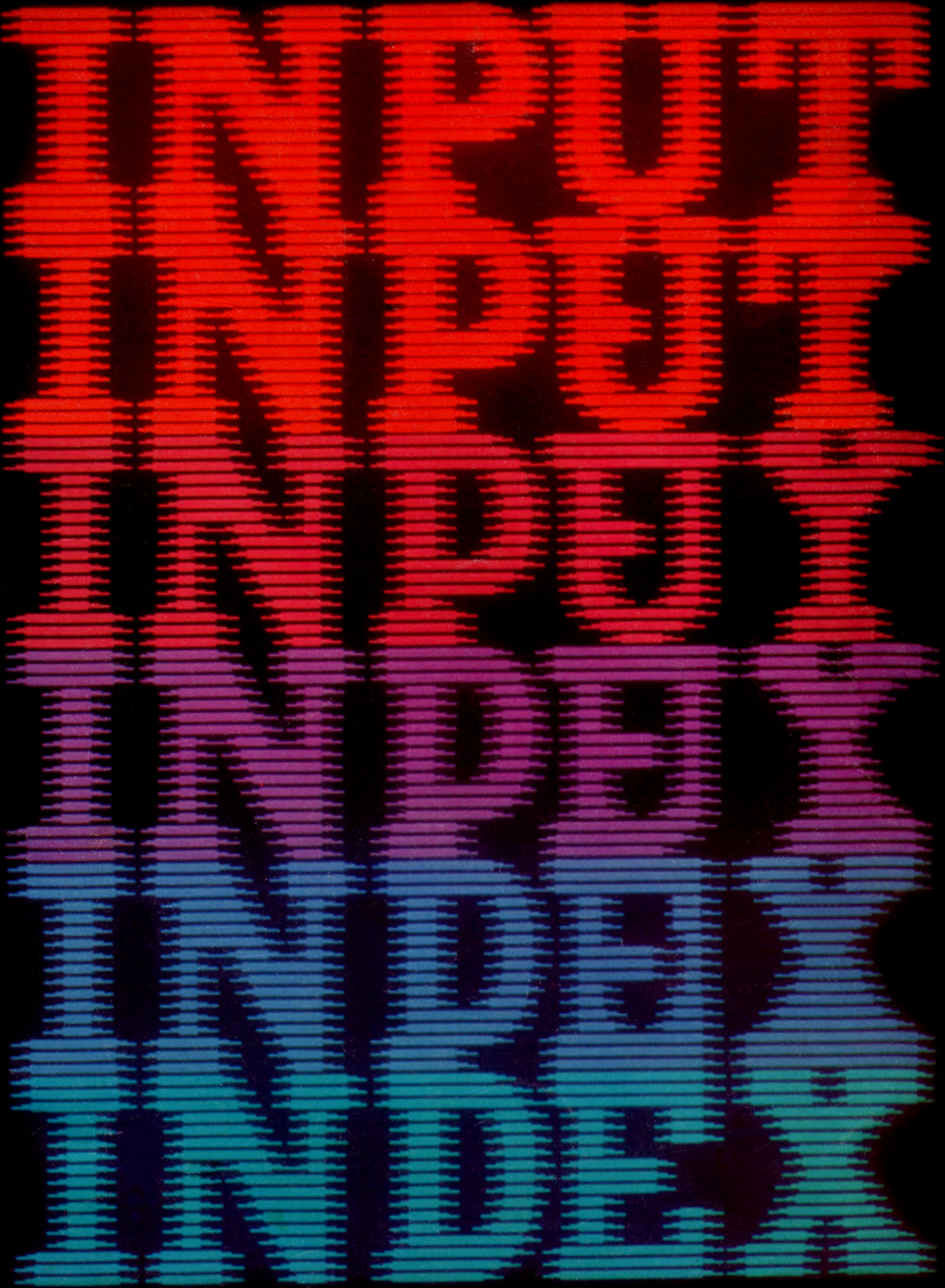
A MARSHALL CAVENDISH 52 COMPUTER COURSE IN WEEKLY PARTS



UK £1.00 Republic of Ireland £1.25 Malta 85c Australia $2.25 New Zealand $2.95

INPUT

Vol 4 No 52

INDEX

TABLE OF CONTENTS 1622

A guide to all the article titles in each of *INPUT*'s six departments

A–Z INDEX 1628

A full list of all the subjects covered in *INPUT* volumes 1 to 4

CORRECTIONS 1649

Program amendments and updates

To all our readers,

Well, that's it! With this, the complete index to the Input Programming Course, your collection is complete.

It is impossible to predict what the future will bring, but one thing that does seem certain is that computers will play a bigger and bigger part in everyone's life. We hope that you have enjoyed your start in programming and that you have had as much fun reading *INPUT* and trying the programs as we have done in writing it for you.

Whether you have typed in all the programs yet or not, you will still want to keep *INPUT* as a source of reference material for the future. If you haven't bound your copies yet, and want to keep them in tip-top condition, the inside back cover gives details of a last-chance offer to obtain the special *INPUT* binders.

Best wishes for the future in computing.

Andrew Kemp

INPUT STAFF LIST

Editor
Andrew Kemp

Art Editor
Maggi Howells

Deputy Editor
Jon Kirkwood

Technical Editor
Sheila Snowden

Sub Editors
Nigel Cawthorne
Beryl Leitch
David Lester
Lloyd Lindo
Dave Rosam
John Ward

Senior Designers
Paul Bickerstaff
David Copsey
Anita Ruddell

Designers
Chris Mijnheer
David Rowley
Phil Tristram

Programmers
Ian Beynon
Trevor Harwood
John Mitchell
Ravi Sahadevan
Chris Smith

Production Controller
Enid Broderick

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA. England. Typeset by MS Filmsetting Limited, Frome, Somerset. Printed by Cooper Clegg Web Offset Ltd, Gloucester and Howard Hunt Litho, London.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland:
Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd,
Department 980, Newtown Road,
Hove, Sussex BN3 7DN

Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015

New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington

Malta: Binders are available from local newsagents.

There are four binders each holding 13 issues.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).

UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services,
Newtown Road, Hove BN3 7DN

Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:

Subscription Department, Marshall Cavendish Services Ltd,
Newtown Road, Hove, Sussex BN3 7DN

Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to:
Marshall Cavendish Partworks Ltd.

© Marshall Cavendish Limited 1985/6/7
All worldwide rights reserved.

The contents of this publication including software, codes, listings, graphics, illustrations and text are the exclusive property and copyright of Marshall Cavendish Limited and may not be copied, reproduced, transmitted, hired, lent, distributed, stored or modified in any form whatsoever without the prior approval of the Copyright holder.

INPUT IS SPECIALLY DESIGNED FOR:

The SINCLAIR ZX SPECTRUM (16K and 48K), COMMODORE 64, ACORN ELECTRON and BBC B, and the DRAGON 32.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM
COMMODORE 64
ACORN ELECTRON and BBC B
DRAGON 32
ZX81
VIC 20
TANDY TRS80 COLOUR COMPUTER

HOW TO USE THE INDEX

This index contains full details of all the articles and programs appearing in *INPUT*, Parts 1 to 51. To help you to find a particular reference as easily as possible, this information is arranged in two ways, in separate lists.

The first list is on pages 1622 to 1627. Here, you'll find a full table of all the articles arranged in the sections under which they appear in *INPUT*. So all the Games Programs,for example, are listed together. In each section, the article's title is given, followed by a short description of what it contains. So whether you can remember the title of the article, or just what it waş about, this quick-reference guide should find it for you with the minimum of fuss.

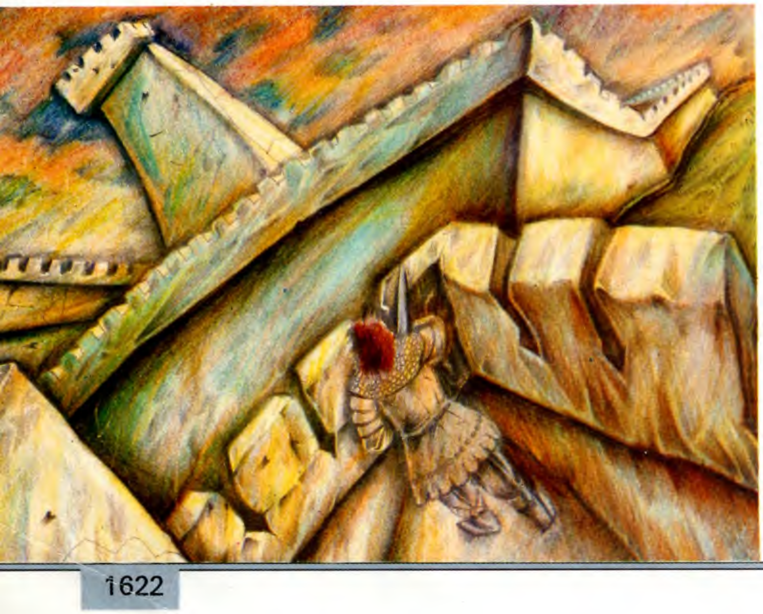
If you are looking for a more specific reference, rather than a whole article, the second index listing on pages 1628 to 1649 is the place to check. This is arranged alphabetically, like the mini-indexes which appear in each issue of *INPUT*. All the main topics are listed separately, and you'll also find references to each of the articles and programs.

When you have found the page reference you are looking for, there's a panel on each page spread which will show you where this appears, by Volume and Part number.

Following the index listings, there is a list of corrections and updates to programs appearing in all four volumes of *INPUT*. This supersedes the occasional errata columns which appear in earlier issues.

1 GET MOVING ON ANIMATION 26–32
Animating ROM graphics
2 RIGHT . . . UP . . . LEFT . . . FIRE! 54–59
Using GET$ and INKEY$ to fire a missile.
3 FUN-TO-PROGRAM MAZE GAMES 68–74
Moving an eater through your own maze design.
4 SCORING AND TIMING ROUTINES 97–103
Time your way through a minefield game
5 DEADLY ENEMIES AND ALIENS 144–151
A space game with aliens, missiles and shields.
6 CREATING THE BIG BANG 161–167
Adding screen explosion UDGs to your games.
7 CREATING LEVELS OF DIFFICULTY 193–200
A new maze game—and how to make it harder.
8 BREAKING THE SOUND BARRIER 230–235
Sound effects for your games.
9 PLANNING AN ADVENTURE 264–268
An introduction to writing your own adventure game.
10 MAPPING OUT AN ADVENTURE 296–301
Planning the story.
11 A MOVING ADVENTURE 344–352
Moving round the locations.
12 THE OBJECTS OF THE QUEST 360–365
Placing objects and writing verb routines.
13 COMPLETING THE ADVENTURE 385–391
Instructions, hazards and warnings.
14 ADVENTURES—THE NEXT STEP 422–427
Now plan you own adventure.
15 PROGRAMMING FOR JOYSTICKS 464–469
A gunsight program for joysticks
16 A DUCK SHOOTING GAME 492–497
Use your joystick to hunt ducks.
17 WHEELING AND DEALING 534–540
Part I of Pontoon card game—the graphics and shuffling.
18 LET PLAY COMMENCE 553–559
Part II of Pontoon—the player's turn.
19 OVER TO THE BANKER 598–604
Part III of Pontoon—the computer's turn.

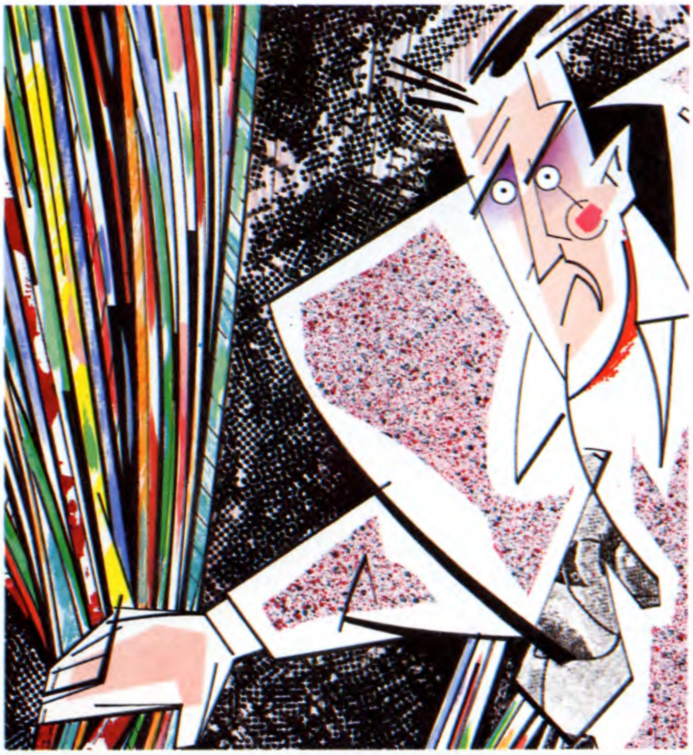
Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620



20 TURN YOUR ADVENTURE INTO AN EPIC 628–636
Assembly language text-compressor for adventure games.
21 DECODING YOUR EPIC 648–655
Decode routine of text-compressor.
22 USING YOUR TEXT-COMPRESSOR 684–689
Encoding, decoding and merging.
23 ON-SCREEN FLIGHT SIMULATOR 716–720
Drawing the cockpit.
24 GET OFF TO A FLYING START 733–739
Setting the aeroplane in motion.
25 THE FINAL APPROACH 765–769
Using the keyboard for control of the flight simulator.
26 SNAKES AND ADDERS 804–810
A number-eating snake game.
27 AS GOOD AS GOLD 830–837
Business strategy game in which you run a mining company.
28 THE MIDAS TOUCH 864–871
Part II of the gold mining game.
29 MARK MY WORDS 899–903
Part I of educational word game.
30 GETTING IT WORD-PERFECT 940–945
Part II of word game.
31 CONTROLLING THE BOARD 980–984
Part I of Othello strategy board game.
32 CONTROLLING THE BOARD—2 1005–1009
Part II of Othello.
33 ARMING THE BANDIT 1028–1033
Part I of fruit machine game.
34 THE FRUITS OF YOUR LABOUR 1074–1080
Part II of fruit machine game.
35 LUNAR TOUCHDOWN 1088–1090
A complete arcade-type game.
36 THE GAME OF FOX AND GEESE 1096–1100
Part I of game using artificial intelligence.
37 FOX AND GEESE GAME—2 1113–1117
Part II.
38 FOX AND GEESE GAME—3 1152–1157
Part III.
39 WILL YOU WALK INTO MY PARLOUR 1177–1183
Part I of Freddy and the Spider from Mars game.
40 GO OUT WITH A BANG 1230–1236
Part II of Freddy and the Spider from Mars.
41 MACHINE CODE LIFE GAME 1237–1239
Simulate the struggle of uni-cellular organisms.
42 WARGAMING: THE FIRST STEPS 1254–1257
Part I of Cavendish Field war game—
43 WARGAMING: OF MAPS AND MEN 1282–1288
Part II of Cavendish Field.
44 WARGAMING: THE ART OF COMMAND 1301–1307
Part III of Cavendish Field.
45 WARGAMING: INTO BATTLE 1346–1351
Part IV of Cavendish Field
46 DESPERATE DECORATOR 1314–1316
A game to catch the paint drips!
47 'MATCH THAT': A COMPUTER PUZZLE. 1356–1357
A colour code guessing game.
48 WARGAMING: MILITARY INTELLIGENCE 1372–1377
Part V of Cavendish Field.
49 ESCAPE: A NEW ADVENTURE GAME 1424–1428
50 ESCAPE: BUILDING UP THE ADVENTURE 1450–1455
51 ESCAPE: THE ADVENTURE GOES ON 1486–1492
52 ESCAPE: ADDING TO THE ADVENTURE 1493–1499
53 ESCAPE: THE CODED TEXT 1545–1551
54 ESCAPE: THE COMPLETE ADVENTURE 1568–1575
55 TUMBLING DICE 1589–1595
Play 'Yacht' a traditional dice game.

1 SPEED UP YOUR GAMES ROUTINES 8–15
Frog and tank UDGs—how to set up and move them.
2 10-MINUTES GAMES GRAPHICS 38–45
Binary/decimal/hex conversions and some more UDGs.
3 GET DOWN TO LOW LEVEL LANGUAGES 65–67
What is machine code and assembly language?
4 DRAW A FIRE-BREATHING DRAGON 80–83
Using UDGs for a useful character.
5 LEARNING TO COUNT ON ONE FINGER 110–116
Understanding binary.
6 HANDLING HEXADECIMAL ARITHMETIC 156–160
Program to convert binary to hex.
7 GETTING DOWN BELOW ZERO 179–183
Negative numbers in hex and binary.
8 MEMORIES ARE MADE OF THIS 208–215
ROM, RAM and memory maps.
9 GETTING TO THE HEART OF IT 236–239
The CPU, the stack and the registers.
10 HOW TO ENTER MACHINE CODE 276–283
Machine code monitor program.
11 ASSEMBLING BY HAND 309–313
Assembly language, mnemonics and addressing.
12 ASSEMBLING BY HAND—2 321–327
13 SPECTRUM ASSEMBLER PROGRAM 380–384
14 A COMMODORE ASSEMBLER PROGRAM 402–405
15 MOVING PICTURES—VIC 20/ZX 81 428–432
Animated graphics—bike, submarine and alien.
16 A DRAGON/TANDY ASSEMBLER 440–444
17 SPECTRUM TRACE PROGRAM 477–483
Program to locate errors.
18 COMMODORE TRACER 514–519
Program to locate errors.
19 ACORN PROGRAM SQUEEZER 546–552
Machine code utility to compress BASIC programs.
20 ACORN PROGRAM SQUEEZER—2 593–595
21 SPECTRUM MICRODRIVE CONVERTER 616–621
Machine code utility to give Microdrive compatibility.
22 DRAGON/TANDY PROGRAM SQUEEZER 637–641
Machine code utility to compress BASIC programs.
23 COMMODORE DISK CONVERTER 676–682
Machine code utility to convert programs for disk drive.
24 THE SPECTRUM SOUNDS OUT 728–732
BEEP and **out** for sound effects.
25 COMMODORE HI-RES GRAPHICS 748–751
Part I—adding new graphics commands.
26 UNDERSTANDING 'FRAMEPRINT' 784–789
Disassembling the graphics routine from Issue 1
27 ADDING INSTRUCTIONS TO BASIC 844–851
Customizing the BASIC on your micro.
28 COMMODORE HI-RES GRAPHICS 872–877
Part II—more new graphics commands.
29 WATCHING THE INTERRUPTS 896–898
A digital clock timer routine.
30 CLIFFHANGER: A COMPLETE GAME 904–913
Part I—title page.
31 CLIFFHANGER: ADDING INSTRUCTIONS 928–932
Part 2—the instructions screen.
32 CLIFFHANGER: TUNING IN 966–973
Part 3—adding a tune.
33 CLIFFHANGER: BEGINNING GRAPHICS 992–997
Part 4—graphics and merging.
34 CLIFFHANGER: SET THE SCENE 1034–1043
Part 5—setting the scene.
35 CLIFFHANGER: PERILS AND PRIZES 1057–1063
Part 6—perils and rewards.
36 CLIFFHANGER: STARTING OFF 1101–1105
Part 7—initializing routine.
37 CLIFFHANGER: RESETTING VARIABLES 1127–1131
Part 8—synchronizing routine.
38 CLIFFHANGER: KEEPING THE SCORE 1145–1151
Part 9—scoring routine.
39 CLIFFHANGER: ADDING SEAGULLS 1204–1208
Part 10—get the flock flapping.
40 COMMODORE ASSEMBLER UPDATE 1214–1215
41 CLIFFHANGER: THE RISING TIDE 1216–1221
Part 11—set the waters rising.
42 CLIFFHANGER: CLOUDING OVER 1240–1244
Part 12—adding weather.
43 CLIFFHANGER: ROCKY 1 1276–1281
Part 13—rolling boulders 1.
44 CLIFFHANGER: ROCKY 2 1328–1332
Part 14—rolling boulders 2.
45 CLIFFHANGER: STEPPING OUT 1338–1345
Part 15—walking Willie.
46 CLIFFHANGER: THE HIGH JUMP 1378–1385
Part 16—jumping Willie 1.
47 CLIFFHANGER: TAKE A RUNNING JUMP 1402–1409
Part 17—jumping Willie 2.
48 CLIFFHANGER: A SAD DEMISE 1440–1447
Part 18—death, sound and end routines.
49 MUSIC WHILE YOU WORK 1448–1449
Interrupt-driven routine to play background music.
50 CLIFFHANGER: SETTLING THE SCORE 1476–1481
Part 19—Willie scores and speeding up.
51 CLIFFHANGER: SNAKES ALIVE! 1520–1524
Part 20—moving the snakes.
52 CLIFFHANGER: SETTING IT OFF 1537–1544
Part 21—main loop.
53 CLIFFHANGER: CHECKING FOR FIT 1580–1588
Part 22—the hex dump for debugging.
54 COMMODORE HI-RES GRAPHICS—3 1596–1607
The complete hi-res utility.

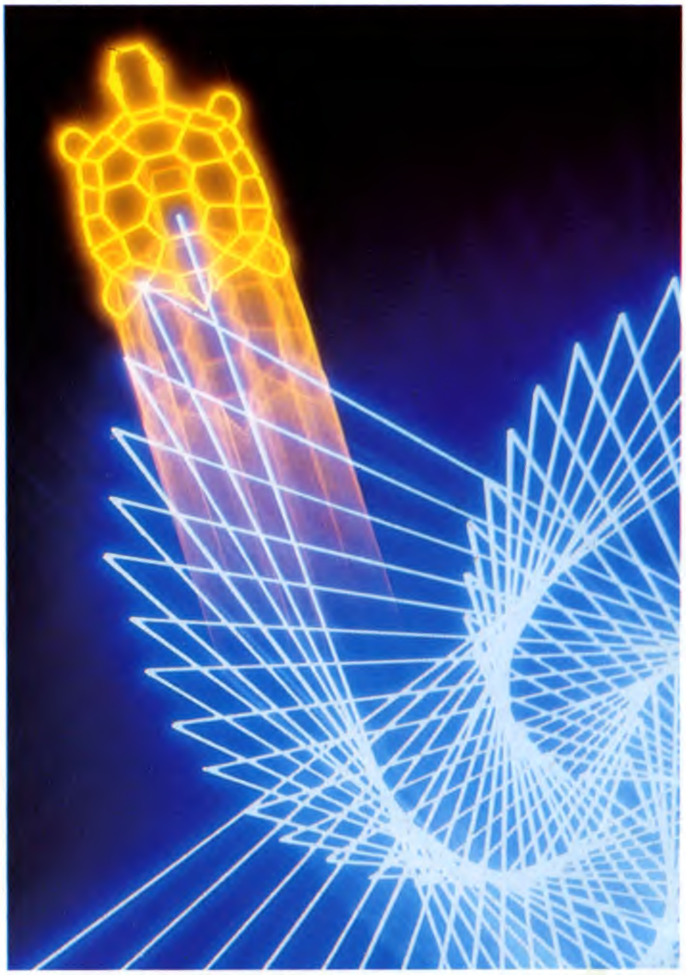
PERIPHERALS



UNTANGLING YOUR SAVES AND LOADS 22–25
Best use of your cassette recorder.
JOYSTICK CONTROLLERS 220–224
Choosing and interfacing a joystick.
PRINTERS: THE CHOICES 225–229
Buying the hardware to take a copy.
THE SPEAKING COMPUTER 398–401
A look at speech synthesizers.
TV VERSUS MONITOR 445–449
Choosing the best VDU.
CHOOSING STORAGE METHODS 504–508
Metits of tape and disk systems.
WHO NEEDS WORDPROCESSING? 541–545
Uses and advantages of wordprocessors.
MODEMS—YOUR LINK TO THE WORLD 612–615
Use your telephone to talk to other computers.
PLUG INTO A PRINTER 642–647
Control commands and escape codes for a printer.
CARING FOR TAPES AND DISKS 683
Indexing and posting your programs and data.
TRIPPING THE LIGHT FANTASTIC 690–693
All about light pens.
ALL ABOUT BULLETIN BOARDS 712–715
Exchanging information between computers.
DATABASE MANAGEMENT SYTEMS 752–757
Manipulating your information stores.
GRAPHICS PADS AND TABLETS 770–774
'Sketching' with your micro.
SETTING UP A DISK DRIVE 820–824
The hardware and the control commands.
COMPUTER-CONTROLLED ROBOTS 884–888
An introduction to robotics.
COMPUTERS IN CONTROL 1552–1556
Use your micro to control mechanical and electrical devices.
MOVING BY DEGREES 1566–1567
Controlling a stepper motor with a BBC.
MUSIC, MICROS AND MIDI 1616–1620
Computer control of synthesizers.

LANGUAGES

1 NEW LANGUAGES: LOOKING AT LOGO 1264–1268
An introduction to LOGO commands and the Turtle.
2 TURNING TURTLE WITH LOGO 1296–1300
Interesting graphics using the Turtle.
3 LOGO: BEYOND THE DRAWING BOARD 1317–1321
LOGO's sprites, maths and wordhandling capabilities.
4 PUTTING TOGETHER PASCAL 1352–1355
The basics of a new language.
5 PATTERNS FOR PASCAL PROGRAMS 1386–1391
Structures of the language.
6 LISP—THE LANGUAGE OF LISTS 1410–1415
List-handling in a new language.
7 CONSTRUCTING A LISP PROGRAM 1456–1460
Defining functions.
8 AND SO ... FORTH 1482–1485
An introduction to a powerful new language.
9 BUILDING UP FORTH 1508–1511
Constants, variables and the contents of the dictionary.
10 THE FORTH DIMENSION 1532–1536
Comparisons, loops and nesting.
11 THE TOWER OF BABEL 1576–1576
The wide world of alternative languages.



Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

APPLICATIONS

1 STREAMLINE YOUR HOBBIES FILE: 1 46–53
Datafile program for hobbies.
2 STREAMLINE YOUR HOBBIES FILE: 2 75–79
Searching, amending and deleting records.
3 MULTIPLE LETTERS MADE EASY 124–128
Letter-writer program incorporating a text-editor.
4 SORT OUT YOUR EXPENSES 136–143
A simple accounting program.
5 DISPLAY YOUR FACTS AND FIGURES 257–263
A bar-chart program.
6 A COMPUTER TYPING TUTOR 289–295
Improve your speed and accuracy at the keyboard.
7 COMPUTER TYPING TUTOR—2 328–332
Part II.
8 COMPUTER TYPING TUTOR—3 353–359
And now a test for you.
9 EXTEND YOUR TYPING 498–503
Practise you new skills on longer sentences.
10 COMPUTER CONVERSION TABLE 520–527
Program to convert imperial to metric.
11 COMPUTER AIDED DESIGN 566–572
Drawing with the cursor.
12 COMPUTER AIDED DESIGN - 2 573–577
Rectangle option and filling with colour.
13 UDGS MADE EASY 721–727
Plotting UDGs on screen and storing them.
14 ADAPTING YOUR UDGS 758–764
Rotate, inverse and mirror routines.
15 FINDING A WAY WITH WORDS 852–856
Part I of a text-editor program.
16 A PLAY ON WORDS 878–883
Part II of the text-editor.
17 GETTING INTO PRINT 914–920
Part III of the text-editor.
18 EXTEND YOUR HOBBIES FILE 946–952
Extra routines for datafile program.
19 LET'S MAKE A DATE 1010–1016
Part I of diary/calendar program.
20 YOURS FOR YEARS TO COME 1017–1021
Part II of diary/calendar program.
21 PLANNING FOR THE FUTURE 1064–1067
Part III of diary/calendar program.
22 LOOKING INTO IT 1081–1087
Program to magnify and reduce your graphics.
23 STARTING WITH SPREADSHEETS 1118–1126
Part I of program to help with your accounting.
24 TAILORING THE SPREADSHEETS 1172–1176
Part II of accounting program.
25 WORKING WITH SPREADSHEETS 1184–1191
Part III—using the program.
26 ASKING THE STARS 1245–1253
A horoscope program you can use as a party game.
27 A COMPUTER INTERIOR DESIGNER 1269–1275
Part I of room-planner program
28 A COMPUTER INTERIOR DESIGNER—2 1308–1313
Part II of room-planner program.
29 MUSIC COMPOSER PROGRAM 1333–1337
Part I.
30 FROM BASIC TO BEETHOVEN 1392–1396
Part II of music composer.
31 FINISHING THE SYMPHONY 1416–1423
Part III of music composer.
32 PLANNING THE BEST COURSE 1429–1433
Project planning using PERT
33 IN SEARCH OF THE BEST TIMES 1466–1473
Putting your PERT program to work.
34 A PICTURE TEST CARD 1474–1475
Check your TV or monitor is giving its best.
35 A PROGRAM CROSS-REFERENCER 1512–1519
A 'search and replace' utility to help with debugging.
36 SPECTRUM/COMMODORE TOOLKIT 1525–1531
A machine code routine to add new commands to BASIC.
37 TUNE IN TO THE WORLD 1562–1565
Tuning in to and decoding radio waves.
38 DISK-EDITING FACILITIES 1608–1615
Retrieve your lost programs.

BASIC PROGRAMMING

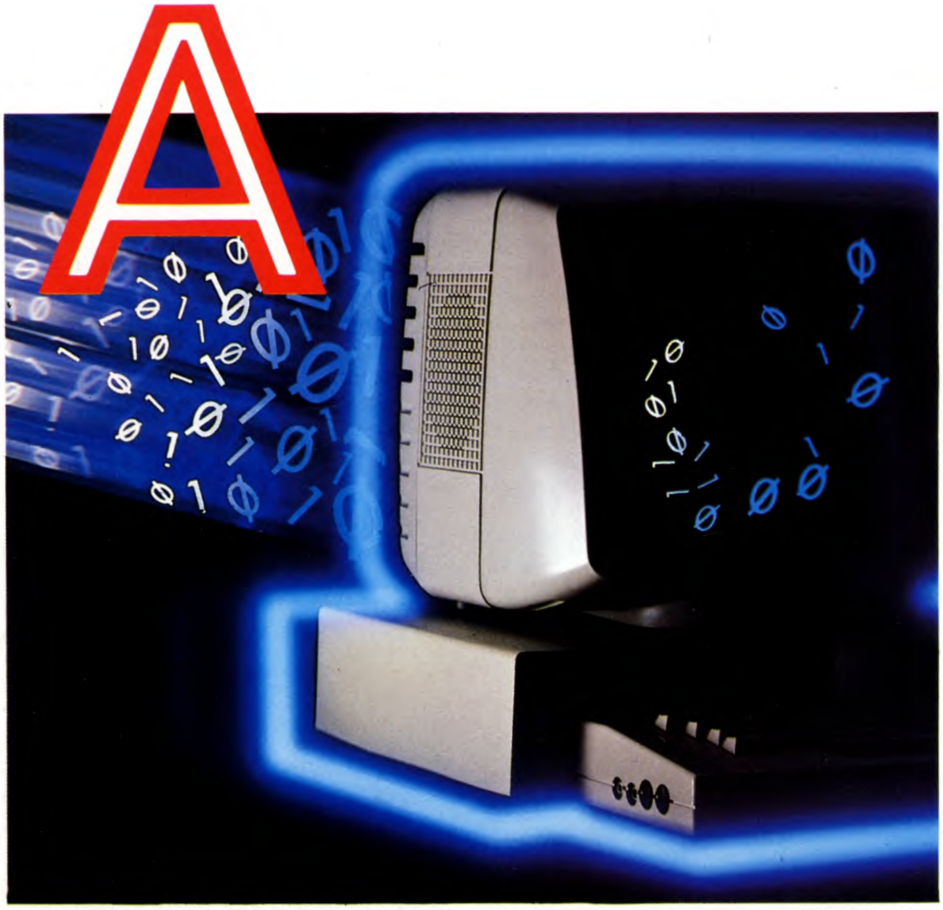
1 THINK OF A NUMBER ... ANY NUMBER 2–7
Using the RND function.
2 THE ART OF THE FOR ... NEXT LOOP 16–21
The computer as a counting device.
3 THE COMPUTER AS DECISION-MAKER 33–37
IF ... THEN commands.
4 THE PROGRAMMER'S ROAD SIGNS 60–64
Forward and backward jumps with GOTO and GOSUB.
5 HOW TO PLOT, DRAW, LINE AND PAINT 84–91
Get started on computer art.
6 VARIABLES: THE MYSTERY EXPLAINED 92–96
Number and string variables and how they are used.
7 ALL ABOUT READ AND DATA 104–109
Storing and gathering information with your micro.
8 SPOT-ON SCREEN DISPLAYS 117–123
Using punctuation for neat screen layout.
9 WHAT DO I DO NEXT? 129–135
Inputting instructions to the computer.
10 ARRAYS—THE INFORMATION STORES 152–155
DIMensioning and using arrays.
11 SPRITES ON THE COMMODORE 64 168–172
An important graphics feature.
12 GET YOUR PROGRAMS IN SHAPE 173–178
Structured programming and the use of flow charts.
13 REFINING YOUR SCREEN GRAPHICS 184–192
Draw, circles and arcs.
14 UNRAVELLING YOUR STRINGS 201–207
Comparing, sorting and slicing strings.
15 GET YOUR PROGRAMS IN SHAPE—2 216–219
Structured programming and a bubble sort routine.
16 UNDERSTANDING PEEK AND POKE 240–247
Looking into the computer's memory.
17 DRAGON/TANDY: BETTER GRAPHICS 248–249
Colour sets and PMODEs.
18 MAKING PICTURES WITH MATHS 250–256
Using SIN and COS, drawing circles, curves and ellipses.
19 CROSS-REFERENCE YOUR ARRAYS 269–275
Multi-dimensional information stores.
20 MEANINGFUL RELATIONSHIPS 284–288
Logical operators—AND, OR and NOT.
21 MORE PICTURES FROM MATHS 302–308
Circular patterns and a working clock.
22 UNDERSTANDING ASCII CODES 314–320
And using them in a code program.
23 SHORTENING PROGRAMS 333
Some tricks to speed up your programs.
24 GETTING RID OF BUGS 334–338
Tracing errors and common faults.
25 HOW TO MERGE PROGRAMS 339–343
Joining programs and adding subroutines.
26 DRAGON/TANDY ANIMATED GRAPHICS 350–352
GET and PUT for easy animation.
27 NEW IDEAS FOR SCREEN ART 366–374
Making more use of colour.
28 AVOIDING PITFALLS 375–379
Buil-in error traps and bombproofing.
29 GETTING THINGS IN ORDER 392–397
Bubble, binary and shell-sort routines.
30 THE POWERS THAT BE 406–412
The power function, squares and cubes.

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

31 HOW TO PLOT GRAPHS 413–419
Making visual displays from your data.
32 COMMODORE KEYBOARD SYMBOLS 420–421
Use the graphics symbols for programming shorthand.
33 IMPROVING YOUR DISPLAYS 433–439
Formatting a neat text page.
34 MAKE MORE OF YOUR UDGS. 450–457
Protecting UDGs in memory and a new character set.
35 PROTECT YOUR PROGRAMS 458–463
Disabling keys, bootstrap programs and copyright statements.
36 HISTOGRAMS AND PIE CHARTS 470–476
More visual displays for your data.
37 PICTURES FROM UDGS 484–491
Creating a jungle scene.
38 IT'S A FRAME-UP 509–513
Wireframe drawings—a grid and a circle.
39 PICTURES FROM UDGS—2 528–533
Finishing the jungle scene.
40 WIREFRAME IN 3-D 560–565
A three-dimensional box graphic,
41 CREATING CUSTOM FUNCTIONS 578–583
Put the DEF FN command to work for you.
42 BOUNCING AROUND IDEAS 584–592
The mathematics of bouncing balls.
43 DRAGON/TANDY PROGRAM EDITING 596–597
Making the most of the edit facilities.
44 GETTING THINGS IN PERSPECTIVE 605–611
Adding perspective and viewpoint to wireframe drawing.
45 CREATING AND USING FILES 622–627
Types of files and how to use them.
46 DETECTING THINGS ON SCREEN 656–661
Detecting collisions with ATTR, PEEK and POINT.

47 WIREFRAMES—ADDING CURVES 662–668
Globes, circles and a space station.
48 SIMPLE MUSIC 669–675
Turn your computer into a keyboard instrument.
49 CHANCE AND PROBABILITY 694–700
Measuring probability with a coin-tossing program.
50 YOU HUM IT—I'LL PLAY IT 701–705
Adding pitch and rhythm to your music.
51 MORE SORTING METHODS 708–711
Delayed replacement, scatter, insertion and quicksorts.
52 WHAT GOES UP MUST COME DOWN 740–747
Programming the path of a flying projectile.
53 USING CONTROL CODES 775
A handy form of programming short-cut.
54 COMMODORE COLOUR SPRITES 776–783
Multi-colour sprite generator program.
55 HOW COMPUTERS STORE NUMBERS 790–796
Floating point arithmetic and numbers in memory.
56 OUT OF THIS WORLD 797–803
Programs to plot trajectories and planetary orbits.
57 MAKING THE HEADLINES 811–819
Enlarging ROM characters and designing your own.
58 COMMODORE/ACORN CUSTOM KEYS 825–829
Programming function keys.
59 MORE HEADLINE IDEAS 838–843
Designing a custom typeface.
60 CONES, CURVES AND CUTS 857–863
Program to explore the attributes of cones.
61 LOOKING INTO CURVES 889–895
Incorporating conic curves into programs.
62 SUPERCHARGE YOUR BASIC 921–927
Speeding up your BASIC programs.

63 ENGINEERING A SOLUTION 933–939
The principles of mechanics and some programs to demonstrate them.
64 EXPLORING THE ACORN'S PAINTBOX 953–959
Colour filling and mixing commands of BBC BASIC.
65 SENDING SECRET MESSAGES 960–965
Secret codes and ciphers and their applications.
66 MULTI-KEY CONTROL 974–979
Understanding the keyboard and a game to use simultaneous keypresses.
67 PLAYING IN HARMONY 985–991
Programming musical harmonies on the Acorn and Commodore 64.
68 PIECING IT TOGETHER 998–1004
Rubber-banding, picking and dragging for easier graphics.
69 'FLICKER-BOOK' ANIMATION 1022–1027
Paged graphics techniques for animation.
70 SENDING SECRET MESSAGES—2 1044–1048
More codes and a code-breaking program.
71 THE MATHEMATICS OF GROWTH 1049–1056
Measuring growth of animals, plants and populations.
72 TELETEXT SCREENS ON THE BBC 1068–1073
How to use this graphics mode.
73 HOW'S THAT SOUND 1091–1095
Explore the technology of digital sound recording.
74 HOW BASIC PROGRAMS ARE STORED 1106–1112
Looking into the BASIC memory stores.
75 MORE ABOUT PAGED GRAPHICS 1132–1137
Extend you animation skills.
76 ENVELOPE SOUNDS 1138–1144
Produce life-like sounds on the Acorn and Commodore 64.
77 PREDICTING THE UNPREDICTABLE 1158–1163
Computing probability.
78 PATTERNS FROM NATURE 1164–1171
Dots, curves and orbits for complex graphics.
79 SOLIDS OF ROTATION 1192–1197
Make three-dimensional shapes.
80 MODELLING REALITY 1198–1203
Predict events in the real world using the RND function.
81 MODELLING: FOOD FOR THOUGHT 1209–1213
Apply modelling methods to a business situation.
82 SQUEEZING OUT A TUNE 1222–1229
Data compression techniques that let you store a tune.
83 USING COMMODORE COLOUR SPRITES 1258–1263
Moving sprites in a complete game.
84 RECURSION—LOOPS WITHIN LOOPS 1289–1295
Understanding one of the key techniques of structures programming.
85 UNDERSTAND THE OPERATING SYSTEM 1322–1327
Accessing OS routines and entry points.
86 FILE IT—DON'T FORGET IT 1358–1364
Saving information out of a program.
87 DRAW IT, PRINT IT 1365–1371
Two screen dump programs for saving graphics to a printer.
88 MODELS OF IRREGULARITY 1397–1401
Use fractals to mimic irregularities.
89 FORMS OF THE NATURAL WORLD 1434–1439
Modelling mountains and snowflakes with fractals.
90 ADDING SOME DEPTH 1461–1465
How to master the tricks of perspective drawing.
91 PAPER, SCISSORS, STONE 1500–1507
A classic bluffing game.
92 PUZZLES, COMPUTERS AND MATHEMATICS 1557–1561
Solving puzzles with your computer.



A

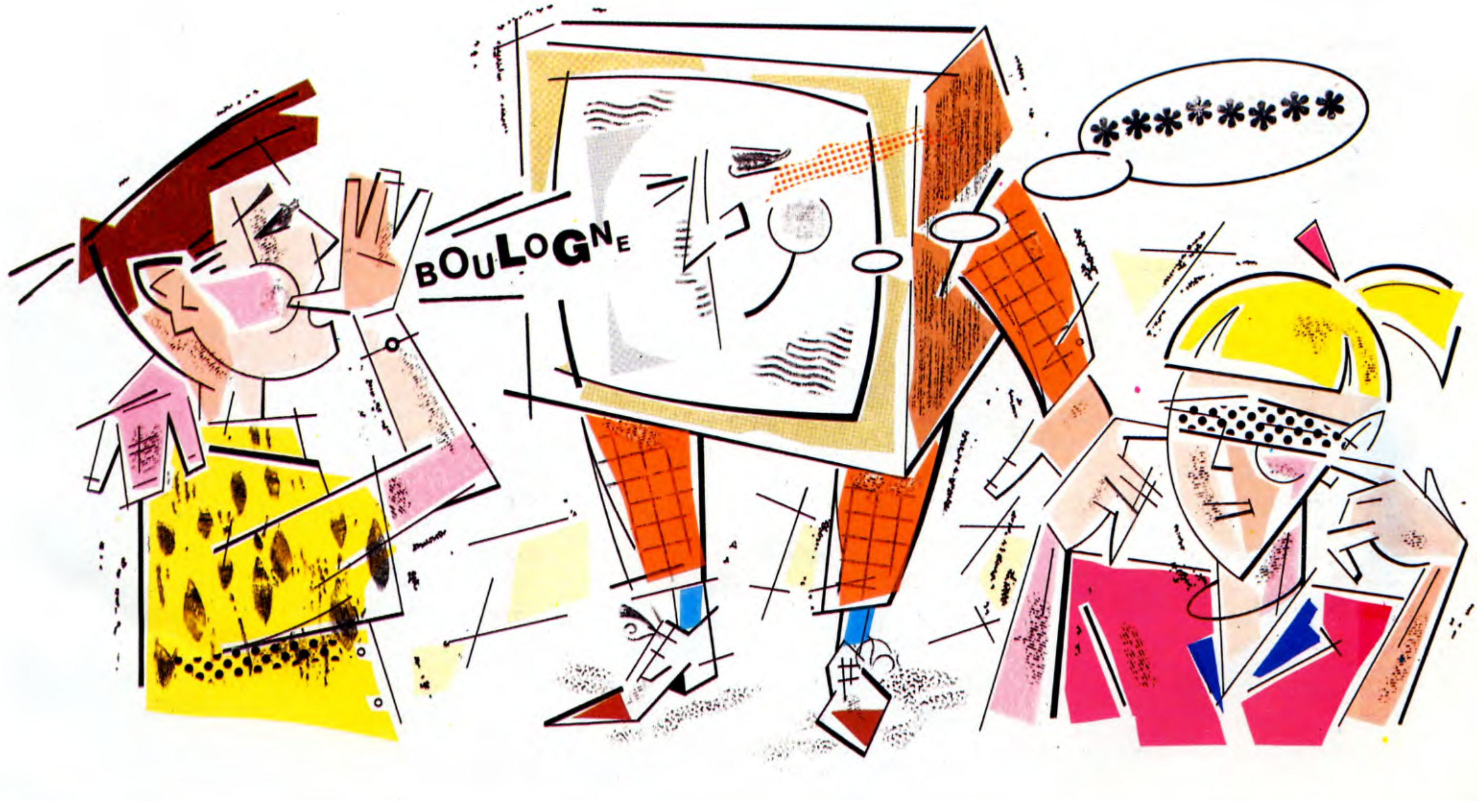
ABS 150, 412
Absolute copying
  in spreadsheet program 1184
Accounting
  see family finance; spreadsheet program
Accumulator
  see registers
Activities, in PERT charts 1430, 1466–1468
Actuators 1555
Addresses, definition 208
Addressing 310–313
  in sideways scrolling routine 322–328
ADSR system
  in sound synthesis
    Acorn 1144
    Commodore 64 1141–1142
Adventure games
  escape 1424–1428, 1450–1455, 1486–1492, 1493–1499, 1545–1551, 1568–1571
  general theory of 264–268
  planning your own 422–427
  quest for the jewelled eyeball 296–301, 344–349, 360–365, 385–391
  text-compressor for 628–636, 684–689
Aeroplane, flying a
  see flight simulator program
Algebra, with LISP 1411
Algol 60 1352, 1576–1578
Algorithms
  alpha-beta 1098, 1113
  in games 1156–1157, 1372–1373
  use of in Pascal 1354, 1389–1390
Aliens
  in games 144–151
  UDG for **ZX81** 430–432
All purpose fill routine
  Acorn 957–959
Allophones
  in speech synthesizers 400–401
Anagrams, program to generate
  Acorn 1326
Analogue models, definition 1198
Analogue sound signal 1092
Analogue-to-digital
  (A-D) converter 722–723
    use of to receive satellite picture
      BBC 1565
AND
  in FORTH 1533
  in graphics programming
    Acorn 371–373
    Spectrum 367
  use of with joystick
    Commodore 64, Vic 20 465–466
ANGL
  Commodore 64 87, 187
@ANGL
  Commodore 64 1596
Angles
  in compass program 251–253
  in clock program 302–306
  of incidence in snooker ball program 587–590
  of trajectory in projectile routines 746–747, 799–800
Animals, measuring growth of 1049–1056
Animation
  of jungle scene 532
  of ROM graphics 26–32
  of solid drawings
    Acorn, Dragon, Spectrum, Tandy 1192–1197
  of sprites
    Commodore 64 1259–1263
  of UDGs
    Vic 20, ZX 81 428–432
  of wireframe drawings 510
  using colour fill techniques
    Acorn 955–959
  using GCOL 3
    Acorn 999–1000
  using paged graphics 1022–1027, 1132–1137
  with LOGO 1317–1320
    also see cliffhanger parts 4 & 10–17; movement
ANSII standard BASIC 1576
Appending programs 339–343
ARC *Commodore 64* 87, 186–187
@ARC *Commodore 64* 1596
ARC COS function
  how to create
    Dragon, Tandy 583
ARC SIN function
  how to create
    Dragon, Tandy 583
Arcade games
  see Freddy and the spider from Mars; lunar touchdown
Arguments, in LISP 1456
Arithmetic
  see binary; decimal; hexadecimal; mathematical functions
Armchair shopping 614
Arrays
  one-dimensional 152–155
  reDIMensioning in adventure games 425–427
  sorting of 216–219
  string
    storing of in memory 1105–1112
    to store text
      Acorn 1325
  two-dimensional 269–275
  troop and map, in war game 1282–1288
  use of in file handling 1358–1364
  use of in LISP 1459–1460
  use of with colour fill techniques
    Acorn 958
Artificial intelligence 884, 1264, 1294
  in Cavendish Field game 1372–1377
  in LISP 1410–1411
    also see fox and geese game
ASCII codes 314–320
  files 622–623
  for extended graphics commands
    Commodore 64 751
  limiting, to compress text 628
  list of, for graphics characters
    Commodore 64 421
  of BASIC programs in memory 1106–1112
  of function keys
    Acorn 829
    Commodore 64, Vic 20 826
  of Teletext graphics
    BBC 1068–1073

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

table of 243
for games 928–932
teletype radio signal 1564
use of in bulletin boards 712–715
use of in codeword program 315–318
with PEEK and POKE 242
Assembler directive, definition 483
Assembler
definition of program 67
Commodore 64 402–405
update of 1214–1215
Dragon, Tandy 440–444
Spectrum 380–384
Assembly language
advantages of 67
by hand 309–313
for sideways scroll routine 321–327
version of UDG frameprint routine 784–789
Assignment statement 92, 152
Astrology program
see horoscope program
Atari joystick, program for 466
Atoms, in LISP 1412–1415
ATTR
Spectrum 68–69, 150, 194–195, 656–658
adding a new instruction 844–847
Attractor set, definition 1170
Attributes area, *Spectrum* 209
Auto
line numbering
Commodore 64, Spectrum 1525–1530
repeat using PEEK and POKE
Commodore 64 246, 976
Dragon, Tandy 244
run 459–461
Averages, working out 33
Axes, in graphs
scaling for bar charts 470–471
setting up 415–416
maximum and minimum values for 419
Axes, screen
in wireframe drawing 560–565, 605–611
use of to plot projectiles 740–747

B

Back-delete techniques
Commodore 64, Vic 20 462
Background music, program to play
Acorn, Commodore 64 1448–1449
Backus Naur form (BNF) 1386
Bandwidth, on TVs and monitors 447
Banks, memory
range of on *Commodore 64* 1258–1259
Bar chart program 257–263, 470–474
BASE, in FORTH 1509
Bases, number
see number bases
BASIC
adding instructions to
Acorn, Dragon, Spectrum 844–851
with toolkit program
Commodore 64, Spectrum 1525–1531
extending graphics capabilities
Commodore 64 748–751
interpreter in memory 209–215
versions of 1576, 1579
Baud rates 713
Beasty, connecting and controlling 887–888
BEEP *Spectrum* 230
increasing volume of 673
use of for Greensleeves tune 966–970
use of to create music 672, 732
BEGIN . . . loops, in FORTH 1534
Bernoulli processes, definition 1198
Betting routine, in Pontoon game 553–559
BFLASH *Commodore 64* 368
Bifurcation tree, definition 1171
Binary
calculations in 113–116
coded-decimal (BCD) 238–239
conversion to decimal 38
using STR$ 205
conversion to hex 38–39
program for 157–160
in DATA for UDGs 40–45
negative numbers 179–183
point 792
power function in 406–407
search routine 926–927
sort routines 396–397
Binomial theorem 696
Bit, definition of 113, 208
flipping of 180–183
masking using AND and OR
Commodore 64 288
Blanking out routine
in maze games 68–74
in space station game 146–151



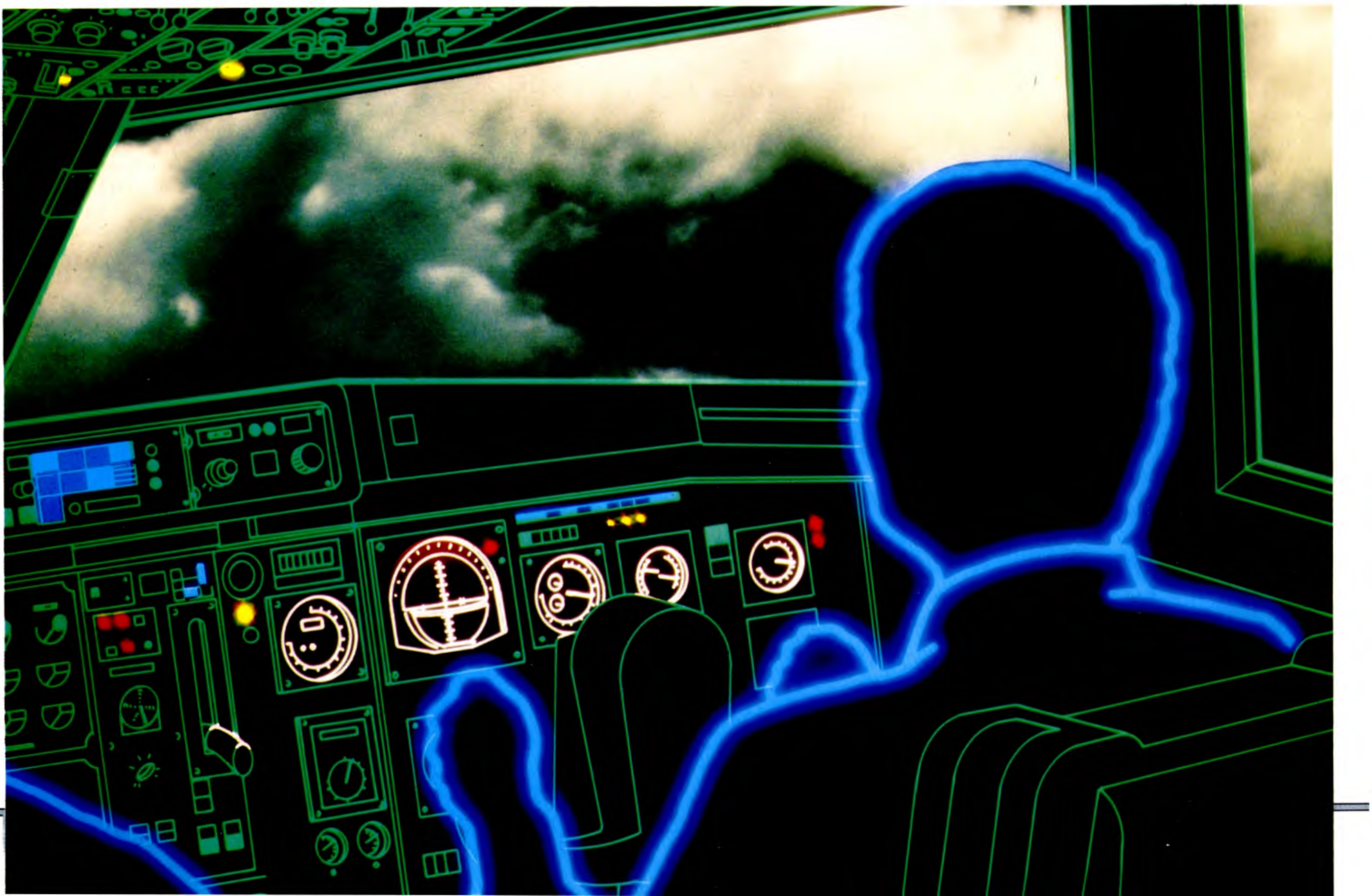
with spaces
Commodore 64 122–123
BLOCK *Commodore 64* 187–188
@BLOCK *Commodore 64*
routine to set up 877
Block, on disks 507, 622
block availability map (BAM), on
1541 disk drive 1610
Block delete, machine code routine for
Commodore 64, Spectrum 1525–1530
Block letters, designing your own 815–819
Bluffing games 1500–1507
Board games
see fox and geese; othello
Body, in FORTH 1484
Boolean variable 1389
Bootstrap programs 459–463
BORDER *Spectrum* 85–86
Border, decorative 1287
Acorn, Dragon, Tandy 205–206
Brackets, use of 35
with LISP 1412–1415
with LOGO 1321
Bridge, drawing a
Spectrum 108
Bubble sort routine 216–219, 392–396
Budgeting
see spreadsheet program
Bugs
getting rid of 334
in formatting 433
notes on debugging 165Ø

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

tracing
on *Dragon, Tandy* 444
with cross-referencer 1512–1519
with hex dump in cliffhanger 1580–1588
with trace programs
Commodore 64, Vic 20 514–519
Spectrum 477–483
also see errors
Bulletin boards 613, 712–715
Business strategy games
see goldmine
BYE *Acorn*
programming a new command 847–849
Byte, definition of 114, 208
arrangement of in sprites
Commodore 64 168
array, in war game 1282–1288
in addressing 310–313
two-byte numbers 237
use of with FORTH 1509–1510

C

Calculator program, for
all number bases 112–113
Calculator stack, definition
Spectrum 210
ZX81 212
Calendar program
par I 1010–1016
part II 1017–1021
part III 1064–1067
CAR, in LISP 1414
Card game program
see pontoon
Cartridge storage 506
CASE structure 212
Case statement, in Pascal 1388
Cassette handling by OS
Dragon, Tandy 1326–1327
Cassette recorders
choice of 23
cleaning 24
Cassette storage 504–505
Castle, drawing a
Dragon, Tandy 109
Casualties, calculating in
war games 1346–1349
CAT command
Acorn 824
Spectrum 820–822
Catastrophe theory 1167
Catenary curve 857
Cathode ray tube, how it works 445–449
Cavendish Field war game
part I—design rules and
setting up UDGs 1254–1257
part II—map and troop arrays 1282–1288
part III—issuing orders 1301–1307
part IV—combat and morale
routines 1346–1351
part V—strengthening the
computer 1372–1377
CCITT 712
CDR, in LISP 1414
Cells,
in animation 1022



in spreadsheets 1119–1120
Channels
opening for data storage 626–627, 1358–1361
CHAR *Commodore 64, Vic 20* 368–370
@CHAR *Commodore 64* 1598
Character generator, *Vic 20* 213
Character sets
Commodore 64 420
designing your own 814–819
enlarging ROM
Acorn, Commodore 64, Spectrum 811–815
redefining with UDGs 450–457
Characters, table of for text compressor 636
Checksums, in DATA statements
use of in desperate decorator game 1314–1316
use of in screen dump 1369–1371
Chips, memory 208
Chords, musical
definition 985–986
programs to play
Acorn, Commodore 64 986–991
CHR$
for low resolution graphics
Dragon, Tandy 26–27
to control cursor
Acorn 43
Ciphers
see codes, secret
CIRCLE command
Commodore 64 87
Dragon, Tandy 91
Spectrum 86, 186
Vic 20 88, 188
use of to draw compass 252–253
Circles
as basis for wirefeame drawing 513
drawing 250–256, 858
with hi-res graphics commands
Commodore 64 1599–1607
option in CAD program 573–577
uses of 863, 893–894
in LOGO 1298
CLEAR command
Dragon, Tandy 278
Spectrum 276
Clear screen, symbols for 4
Cliffhanger game
see separate contents list
Clock,
digital, routine for 896–898
drawing a 24-hour 302–306
internal 69–73, 101–103
CLOSE command
Commodore 64 153
CLR *Commodore 64*
OS routine for 1323
CMD
Commodore 64, Vic 20 644
COBOL language 1578
CODE command
see ASCII
Codebook program 1047–1048
Coded message program 315–318
Codes
ASCII 243, 314–320
control 775
colour, in guessing game 1356–1357

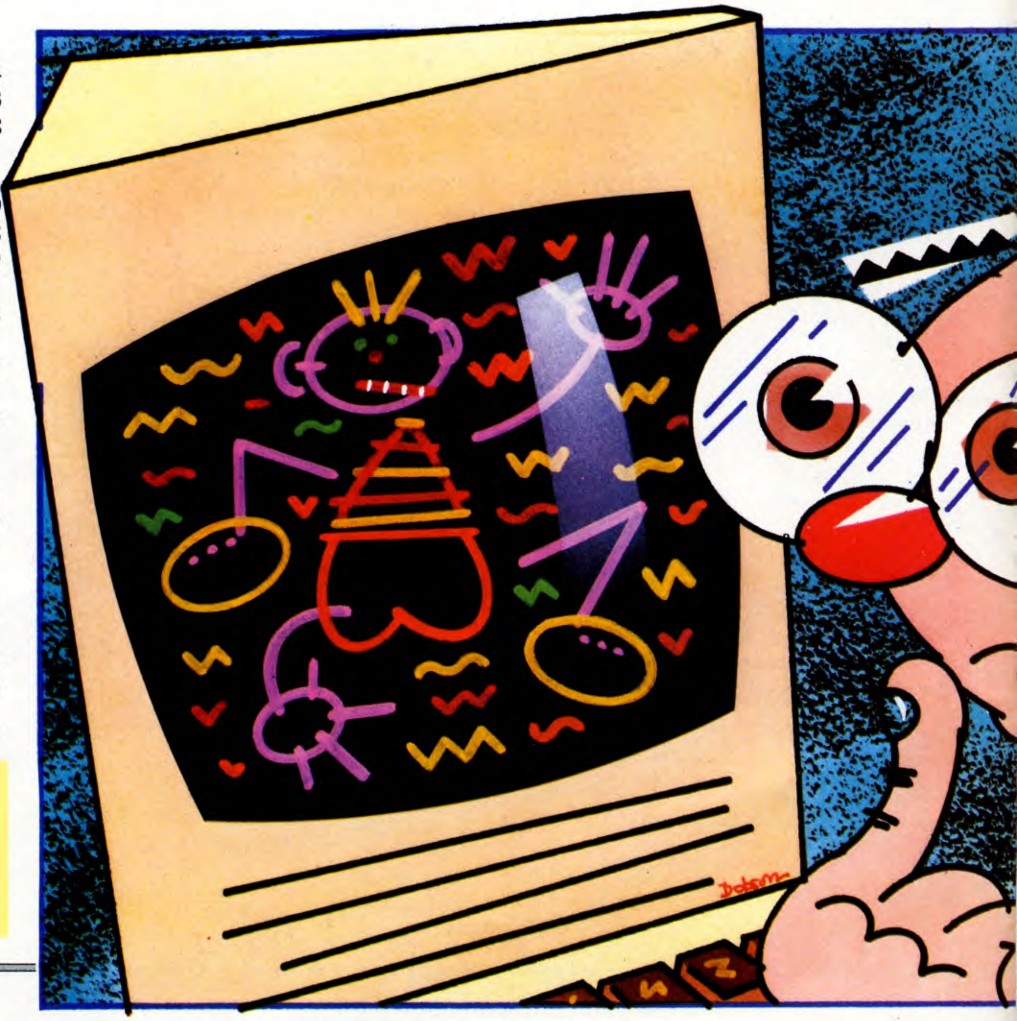
for text-compressor 628–629
routine for decoding 648–655
table of 636
Gray 1566–1567
radio 1562–1564
secret 960–965, 1044–1048
Collisions, detection of
boulders in cliffhanger 1276–1281
programming for 656–661
sprites
Commodore 64 783, 1263
Colon definitions, in FORTH 1510
COLOR command
Dragon, Tandy 374
COLOUR command
Acorn 371
Commodore 64 87
Vic 20 88
@COLOUR
Commodore 64 1596
Colour
avoiding overlapping of
Spectrum 577
code guessing game 1356–1357
detecting using ATTR, PEEK and POINT 656–661
in CAD program 567–572, 573–577
memory
Vic 20 214
representing in tonal screen dump 1369–1371
sets
Dragon, Tandy 249, 352
shading effects 1464–1465
sprites
Commodore 64 776–783, 1262
use of in jungle scene 532
use of in life game 1237–1239
use of in screen displays 433–439
with wireframe drawing 512
Commands, adding to BASIC
Acorn, Dragon, Spectrum 844–851
Commodore 64, Spectrum 1525–1531
Comment lines,
in FORTH 1511
Communications, between databases 612–615, 712–715
Comparisons, in FORTH 1532–1533
Compass, drawing a 251–253
Compiled languages 1353, 1482
Compilers, for Pascal 1390
Compiling programs 66, 926
Computer-Aided-Design (CAD)
program for 566–572, 573–577
rubber-banding and picking and dragging 998–1004
Concatenation 201
also see variables, string
COND, in LISP 1415
Conic sections
program to draw 857–863
using in programs 889–895
CONS, in LISP 1414
Constants
in FORTH 1508–1510
in spreadsheet program 1184
Context vocabulary, in FORTH 1484
Continuous input option
in hobbies file program
Spectrum 947

- **Continuous print option**
  - in hobbies file program
    - *Acorn* 949
    - *Commodore 64* 947
    - *Dragon, Tandy* 951–952
    - *Spectrum* 947
- **Control codes,**
  - ASCII 319–320
    - of function keys
      - *Commodore 64, Vic 20* 826–828
- **Control commands**
  - for disk drives
    - *Acorn, Commodore 64, Dragon, Vic 20* 824
  - for Microdrives
    - *Spectrum* 822
  - for printers 644–647
  - in wordprocessor program 545
- **Control systems** 1552–1556, 1566–1567
- **Control variables**
  - see variables
- **Conversion program**
  - for metric and imperial measurement 520–527
- **COPY command**
  - *Spectrum, ZX81* 644, 1365–1366
- **Copyright statements**
  - *Spectrum* 462
- **COS** 250–256, 302–308
  - in fractal programs 1398–1401
- **Counting**
  - in computerese 237
  - speed of 17
  - using FOR . . . NEXT loops 17
- **CPU**
  - see microprocessor
- **Crash sound**
  - *Acorn* 234
- **Cricket average program** 118–123
- **Critical path** 1429, 1467–1468
- **Crocodile UDG**
  - see jungle scene
- **Cross-referencer utility** 1512–1519
- **Cryptography** 960–965, 1044–1048
- **CSET** *Commodore 64* 369
- **@CSET** *Commodore 64*
  - routine to set up 872
- **CTRL key**
  - *Acorn* 775
- **Cube**
  - drawing a wireframe 560–565
  - function
    - see user-defined functions
  - rotating a wireframe 605–611
- **Cumulative sum technique** 1502, 1505–1507
- **Cursor**
  - control codes for
    - *Commodore 64, Vic 20* 123, 420–421
  - control of
    - *Dragon, Tandy* 122
  - definition of 3
  - drawing with in CAD program 566–572, 573–577
  - editing with
    - *Dragon, Tandy* 597
  - positioning graphics and text
    - *Acorn* 1061
  - routine for wireframe drawing 511
  - use of for better screen displays 434–435
  - use of in magnification program 1081–1087
  - use of in UDG generator 721–727
- **Curves**
  - in fractal programs 1398–1401
  - patterns from 1164–1171
  - program to draw 857–863
  - using in programs 889–895
- **Cusps,** of orbits 1167

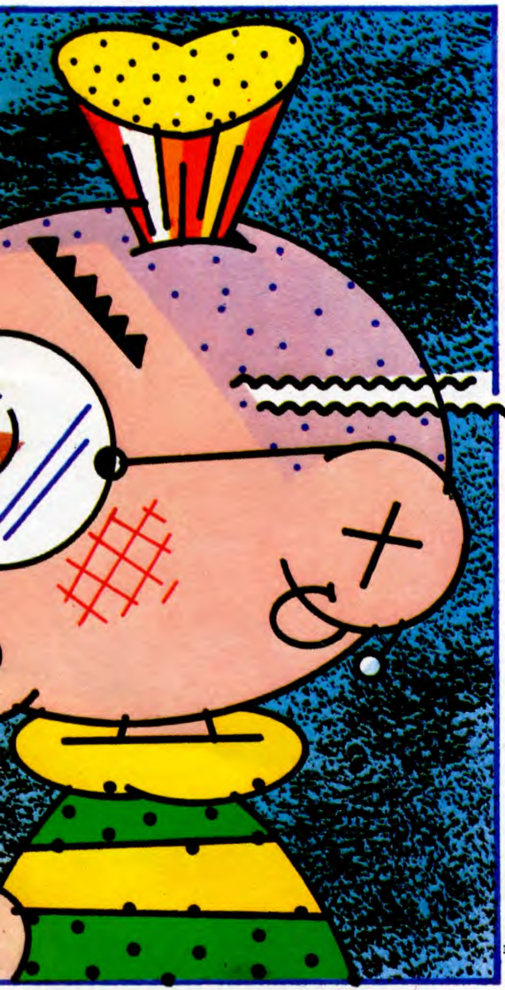
D

- **Daisy chain,** of peripherals
  - *Commodore 64* 647, 825
- **Daisy-wheel printers** 1365
- **DATA statements**
  - best systems for 45
  - changing in typing tutor 332
  - comparison of 1222–1229
  - errors in 337
  - for custom typeface 838–843
  - for graphics 107
  - for music programming 704–707
  - for typing course 499
  - generation program for 169–171
  - in adventure games 360–365
  - in arrays 154–155
  - keeping track of 106
  - printing out from UDG bank 759–764
  - READing 104–109
  - storage of the UDG bank 451–457
  - tables of, in cliffhanger 906–913
  - to define mazes 68–74
  - to define UDGs 10–15, 38, 40–45
  - to elaborate on clock program 306
- **Data**
  - Direction Register (DDR) 884
  - presentation, in graphs 413–419
  - storage
    - see bar charts; DBMS; disk storage; files; pie charts; tape storage
  - transfer 622–627, 712–715
- **Databases** 622–627, 752–757
  - also see arrays, two-dimensional
- **Datafiles**
  - definition of 623
  - program, adding options to 947–952
  - use of in text-editor 852
    - also see files
- **DBMS** 752–757

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620



Debug loop
in machine code routine
Commodore 64 908
Debugging
escape adventure game 1571
cliffhanger game 1580–1588
also see errors
Decimal
conversion to and from
binary 38, 45, 114–115, 205
definition of 110
conversion to and from
hexadecimal 157–160, 593–595
Commodore 64, Spectrum 1527–1530
memory addresses 208–214
Decision making 33–37, 175–178
with logical operators 285–288
Decoding radio signals 1562–1564
DEF FN Command 573–583
Acorn 217
Dragon, Tandy 145
use of for speed 925
Definition, in FORTH
1484, 1510–1511
DEFPROC
see procedures
DEFUN, in LISP 1456–1459
DEFUSR
Dragon, Tandy 283
Delayed replacement sort
routine 708–710
Delaying action in programs
using FOR . . . NEXT loops 17
using GET$ and INKEY$ 134–135
also see cliffhanger parts 8 & 11



Design program
see Computer Aided Design
Design specification, in
program structuring 174
Desperate decorator game 1314–1316
Device number, for datafile
Commodore 64 1360
DFS
Acorn 821, 1609
Diagonal element, in puzzle solving 1560
Dialects of FORTH 1535–1536
Diary program
part I 1010–1016
part II 1917–1021
part III 1064–1067
Dice-throwing 63–64
calculating probabilities 700
yacht game 1589–1595
Dictionary, in FORTH
1482, 1510–1511
Digital clock routine 896–898
Digital sound signal 1092
Digital tracers 771–772
Digitalization 772
DIM statement 152–154
to create memory space
ZX81 276
with GET
Dragon, Tandy 351
also see arrays
Dimensions, mathematical 1398
Diminishing sizes, in perspective
drawing 1463–1464
DIN plugs 22–25
for MIDI 1619
Dip switches 646
DIR command
Dragon 824
Direct access file 754
Direct entry control codes,
table of *BBC* 1073
Direct page, *Dragon, Tandy* 215
use of in addressing 312–313
Directory, of disk format 1608–1614
Directory program 105
Directrix, definition of 890
Disabling keys 379
to protect programs
Dragon, Spectrum, Tandy 462–463
Disassembling UDG grid routine 784–789
Disk
drive option, in hobbies file program
Acorn, Commodore 64, Dragon 949–952
format map
Acorn 1612–1613
Commodore 64 1610
Dragon, Tandy 1614
monitor utility, to retrieve and
amend stored information 1608–1615
SAVEing to and LOADing from
Commodore 64 676–682
storage
advantages of 544–545
files for 622
protection of 683
Display area
Spectrum 209
Display file
ZX81 212
Distance codes program 960–962
Distribution curves, in coin

tossing program 697–700
DMON utility 1608–1615
DO . . . loop, in FORTH
1533–1534
Doppler effect 1140
DOS 1609
Dot command, in FORTH 1485
Dot-matrix printer 1365
Dot patterns 1167–1170
Dotted pairs, in LISP 1460
Double density coordinates
Commodore 64 1129, 1279, 1341
Double precision integers,
in FORTH 1509
Downloading 614, 712–715
Dragon curve, drawing with
fractals 1400–1401
Dragon, fire-breathing 81–83
DRAW command
absolute, how to create
Spectrum 1324
for colour car
Spectrum 367–368
in golf course scene
Acorn, Dragon,
Spectrum, Tandy 184–192
in graph program 254–255
in random maze game
Dragon, Tandy 199–200
in snow scene
Commodore 64 188
in space station game
Dragon, Tandy 145
to create UDGs in war game
Dragon, Tandy 1254–1256
@DRAW *Commodore 64* 1596–1598
Drawing
a new typeface 838–843
aids
see CAD program; graphics pads;
light pens
letter
Dragon, Tandy 191–192

  in magnification program 1081–1087
  in room planner program 1269–1275, 1308–1313
  perspective 1461–1465
  using GET, GET$ and INKEY$ 132–133
  with LOGO 1296–1300
Drum machines 1618
Duck-shooting game 492–497
Duel program
  Acorn, Dragon, Spectrum 977–979
Dumb software 714
Dummy numbers 34, 1364
Dummy variables
  see parameters
Dumping
  see screen dump
Duration, in music 704
Dynamics, programs to illustrate 1164–1171

E

Eccentricity, of ellipses 800–801, 895
Editing
  area
    Spectrum 210
    ZX81 212
  commands
    Commodore 64 420–421
    Dragon, Tandy 597
  facility on assembler program
    Commodore 64 1214–1215
  information stored on disk 1608–1615
  printouts
    see wordprocessing
  sprites in generator program
    Commodore 64 777–780
  UDGs in generator program 721–727, 758–764
  using function keys
    Acorn 829
  with cross-referencer 1512–1519
  with LOGO 1296
  with Pascal 1355, 1391
  with text-editor program 852–856, 878–883, 914–920
Education by computer
  see LOGO
Educational games
  see wordgame
Egg-timer programs 176–177, 955–956

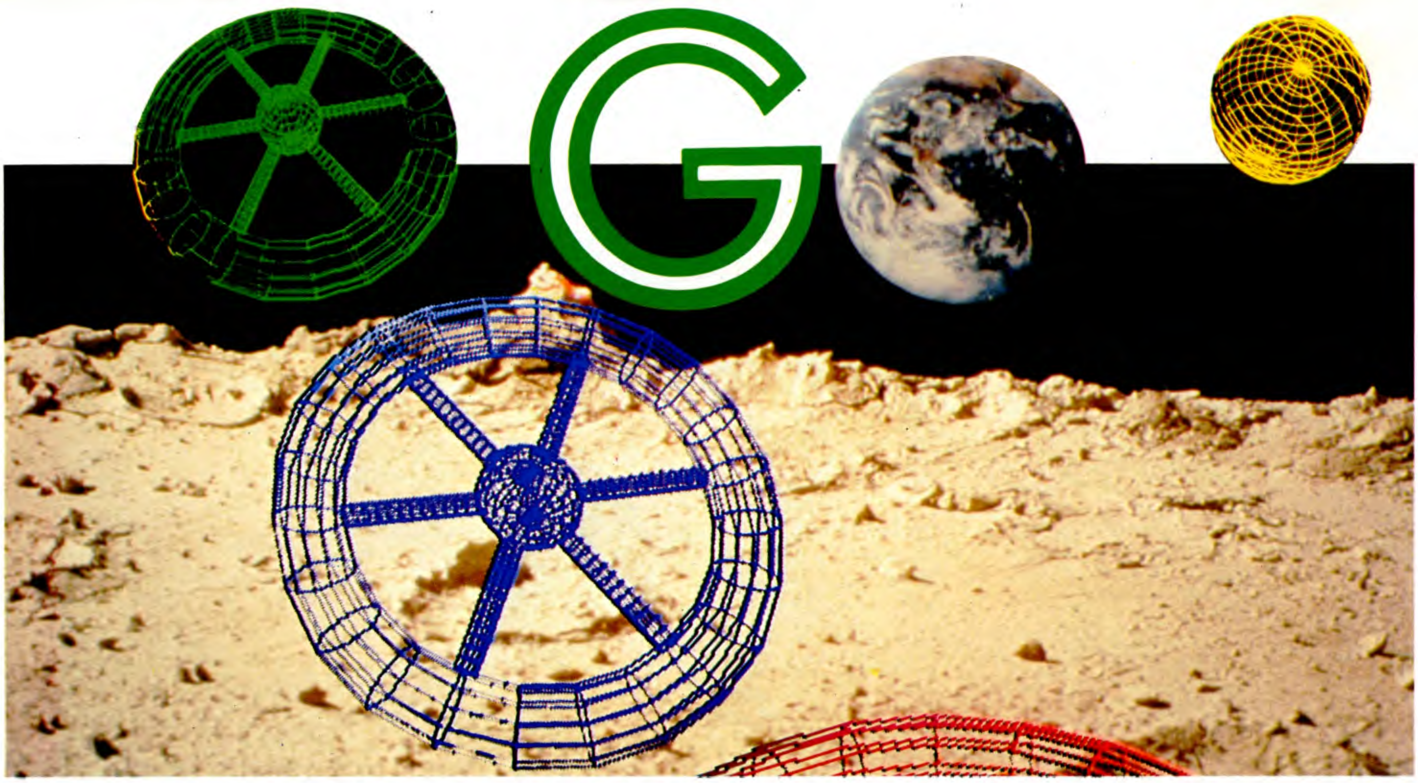
Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

Electronic mail 614
Elephant UDG
  see jungle scene
ELIZA 1410–1411
Ellipses
  creating with DEF FN
    Dragon, Tandy 581
  drawing with hi-res graphics commands
    Commodore 64 1599–1607
  in CAD program 574–577
  in pattern program 894–895
  in slipping ladder program 890–891
  program to draw 858–859
  uses of 863
EMIT, in FORTH 1510
END 105–106
End-of-file markers 1363–1364
ENDPROC
  see procedures
Engineering
  see mechanics, principles of
Enlarging a typeface 838–843
Enlarging sprites
  Commodore 64 1262
Entry, of datafile 624
Envelopes
  of orbits 1164–1167
  sound *Acorn,*
    Commodore 64 232–234, 707
    amplitude 1139–1140
    combined *BBC* 1144
    in music composer program 1417–1419
    in musical harmony program 986–991
    loud and quiet in cliffhanger 1243–1244, 1442–1446
    parameters of 968–971, 1138–1139
      program to change 1142–1144
    pitch 1140, 1143
EOF
  Acorn, Dragon, Tandy 1363–1364
EOR *Acorn* 287
  in graphics programming 371–372
EPROM 506
Epson printers 1366
  codes for 646–647
EQ, in LISP 1415
Equality testing, in LISP 1458–1459
Equations, mathematical
  program to illustrate 584–592
  simultaneous and singular 1559
ERASE
  Spectrum 822
Errors
  causes of 36, 58
  checking for in DATA 490
  correcting of in CAD program
    Acorn, Dragon, Tandy 577
    Commodore 64, Vic 20 576
    Spectrum 574
  correcting with text-editor 852–856, 878–883
  correcting with wordprocessor 542
  corruption of variables 216
  in long programs 77
    Commodore 64 405
    Spectrum 383
  messages
    in *Spectrum* assembler 384
    program for 378–379
    table of 336
  safeguarding against 375–379
  tracing
    in cliffhanger with
      hex dump 1580–1588
    Commodore 64 514–519
    Spectrum 477–483
  trapping 951
  typing, how to delete 498–503
  with READ and DATA 105
    also see bugs; editing
Escape adventure game
  part I 1424–1428
  part II 1450–1455
  part III 1486–1492
  part IV 1493–1499
  part V 1545–1551
  part VI 1568–1575
Escape codes for printers 646–647
EVAL, in LISP 1459
Evaluation, in LISP 1412–1415
Events, in PERT charts 1430, 1466–1467

- **EXEC** *Dragon, Tandy* 283
  - to access OS 1326–1327
- **Exits,** in adventure games 298–300, 346, 349
- **EXPECT,** in FORTH 1511
- **Expert Systems** 1411
- **Exploding the memory**
  - *Acorn* 491
- **Explosions**
  - sounds for
    - *Acorn, Dragon, Tandy* 234–235
  - visual 161–167
- **Exponential distribution**
  - of random variables, program to show 1202
- **Exponential smoothing technique** 1502–1505
- **Exponents** 790–796
- **Exporting data** 754
- **Expressions,** in LISP 1412

F

- **Factorials,** calculating
  - BASIC program for 1291–1293
  - in LISP 1458–1459
- **Falling object program** 411–412
- **Family finance program** 136–143
- **Fence post code** 1046
- **Fetching and storing,** in FORTH 1509–1510
- **Fibonacci numbers** 1056
- **Fields**
  - adding to hobbies file 949–951
  - definition of 46
  - in datafiles 624
  - in DBMS 756–757
- **Files** 622–625
  - amending on disk 1608–1615
  - creating 1358–1361
  - in DBMS 752–757
  - input/output of 626–627
  - marking end of 1363–1364
  - naming of 624, 826
  - reading 1361–1363
  - sprite
    - *Commodore 64* 780–781
- **Filing system programs** 48–53, 77–79
- **Filing program extra routines** 946–952
- **Fill file** 542–543
- **Filling in with colour**
  - *Acorn* 953–959
- **Fire-breathing dragon** 81–83
- **Firing a missile** 55–57, 146–151
- **Fish population program** 1170–1171
- **Five-key target program** 976–977
- **Flag**
  - definition of 237
  - in bubble sort routine 219
  - in FORTH 1532
- **Flame effects for games** 162–166
- **FLASH**
  - *Commodore 64* 368
  - *Spectrum* 86, 434, 436
- **Flashing**
  - alien UDG, *ZX81* 432
  - effects for screen displays 436–439
  - in Teletext mode
    - *BBC* 1070
- **Flight simulation program**
  - part 1—drawing the cockpit 716–720
  - part II—auto-pilot 733–739
  - part III—pilot's controls 765–769
- **Floating point numbers** 792–795
- **Flow charts,** for program structuring 174–178
- **Focus,** of curves 889–895, 1166–1167
- **Food stall model program** 1209–1213
- **Football results program** 1200–1201
- **FOR . . .,** in Pascal 1387
- **FOR . . . NEXT loop**
  - definition of 16
  - flow chart of 178
  - for delaying action 17
  - for graphics effects 18–21, 40–41, 164, 166, 189
  - for movement of ROM graphics 27–32
  - in two-dimensional arrays 272–275
  - speed of 924
  - to play musical notes
    - *Spectrum* 230–231
  - to READ DATA 105
  - to 'shade', *Spectrum* 85
- **FORGET,** in FORTH 1510
- **Form letter** 542–543
  - in text-editor 914–920
- **Formatting**
  - a disk/cartridge 507, 826
  - improving your screen 433–439
  - numbers on screen
    - *Acorn, Dragon, Tandy* 795–796
  - printouts by wordprocessor 545
  - with text-editor 914–920
- **FORTH**
  - part I—terminology and stack manipulation 1482–1485
  - part II—variables and new definitions 1508–1511
  - part III—comparisons and loops 1532–1536
- **FORTRAN** language 1576–1578
- **Fox and geese game**
  - part I—principles and graphics 1096–1100
  - part II—initializing and mapping the moves 1113–1117
  - part III—higher levels 1152–1157
- **Fractals** 1397–1401, 1434–1439
- **Fractions**
  - conversion of to metric 521
  - program for converting into binary 114–115
- **Frameprint routine for UDGs**
  - assembly language version of 784–789
- **Freddy and the spider from Mars game**
  - part I—the graphics 1177–1183
  - part II—animation 1230–1236
- **Frequency**
  - musical 671
  - setting in music composer
    - *Commodore 64* 1417–1418
- **Frequency control register**
  - *Commodore 64* 232
- **Frequency distribution program,** for code-breaking 1044–1046
- **Frequency Shift Keying (FSK),** or radio waves 1564
- **Frog UDG** 10–15
- **Fruit machine programs** 36–37, 1028–1033, 1074–1080
- **Functions,** user-defined
  - see user-defined functions
- **Function keys,** programming
  - *Acorn, Commodore 64, Vic 20* 826–829
- **Functional languages** 1578–1579
- **Functional programming** 1482
- **Funeral march**
  - *Acorn, Dragon, Tandy* 233–235
- **FX command** *Acorn* 71–72, 457, 646
  - to access OS 1324–1326
  - use of with function keys 829



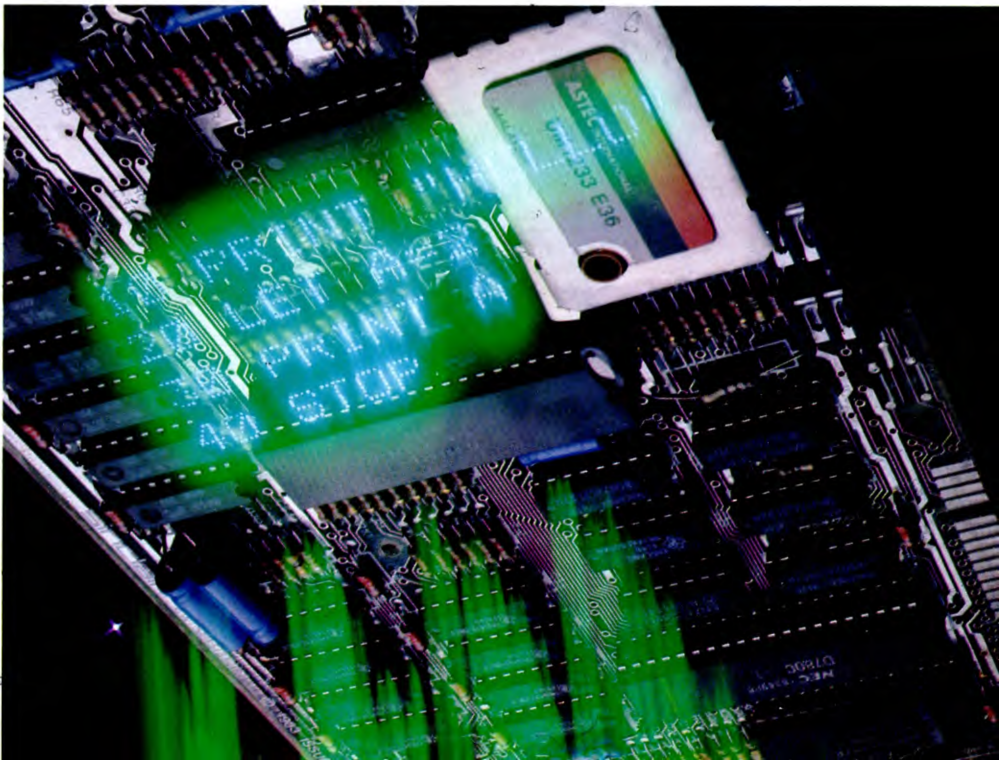
Games
adventure 264–268, 269–301, 344–349, 360–365, 385–391, 422–427, 1424–1428, 1450–1455, 1486–1492, 1493–1499, 1545–1551, 1568–1575
aliens and missiles for 144–151
anagram 202–204
arrays for 155, 1282–1288
bluffing 1500–1507
Cavendish Field war game 1254–1257, 1282–1288, 1301–1307, 1346–1351, 1372–1377
cliffhanger
see contents list
coin-tossing 61–62, 695–700
desperate decorator 1314–1316
dice-throwing 64, 1589–1595
duck-shooting 492–497
escape adventure game 1424–1428, 1450–1455, 1486–1492, 1493–1499, 1545–1551, 1468–1575
firing missiles in 55–58
flight simulation 716–720, 733–739, 765–769
food stall model 1209–1213
fox and geese 1096–1100, 1113–1117, 1152–1157
Freddy and the spider from Mars 1177–1183, 1230–1236
fruit machine 36–37, 1028–1033, 1074–1080
goldmine 830–837, 864–871
guessing 2–7, 34–35, 201–202, 1356–1357
horoscope program 1245–1253
keeping track of moving objects in 656–661
life 1237–1239
lunar touchdown 1088–1090
magnification program 1081–1087
'match that' colour code 1356–1357
maze 68–74, 193–200
minefield 97–103
moving characters in 54–59
multi-key control for 974–979
number-eating snake 804–810
othello 980–984, 1005–1009
paper, scissors, stone 1502–1507
pontoon 534–540, 553–559, 598–604
probability theory in 694–700
quick-draw 103
speed typing 355–359
text-compressor program for 628–636, 684–689
trajectory 797–800
using joysticks with 464–469
visual explosions for 161–167
wordgame 899–903, 940–945
yacht 1589–1595
Garbage collection
in LISP 1460
in LOGO 1299
using EXEC
Dragon, Tandy 1327
GCOL *Acorn* 89, 371–373
GCOL 3 *Acorn*
use of for animation 999–1000
Geometry
fractal 1397–1401, 1434–1439
Turtle 1296
GET command
to animate UDGs
Dragon, Tandy 41, 350–352
to detect keypresses
Commodore 64, Vic 20 55
GET$ *Acorn* 55, 103, 132–135
GET # *Commodore 64, Vic 20* 135
Globe, drawing a wireframe 664–666
Golden ratio 1056
Goldmine game
part I 830–837
part II 864–871
Golf-course, drawing a
Acorn 189–191
Spectrum 184–186
GOSUB 62–64
GOSUB stack in memory
Spectrum, ZX81 211–212
GOTO 60–62
Graphics
adding colour, *Dragon, Tandy* 248–249
bouncing snooker ball program 584–592
colour commands
Acorn 953–959
data statement
BBC 1072
digital tracers for 771–772
direct entry
BBC 1071–1073
displays, programs for
dumping 1365–1371
drawing circles, ellipses and graphs using maths 250–256
drawing 24-hour clock 302–306
effects using curves 857–863, 889–895
elaborate colour effects 366–374
explosions for games 161–167
fire-breathing dragon 80–83
flashing bars
Acorn, Commodore 64, Dragon, Tandy 163–166
frog UDG 10–15
gold course scene
Acorn, Spectrum 184–191
hi-res
combined with UDGs 485–489, 531
for custom typeface 838–843
pseudo
ZX81 432
setting up new commands
Commodore 64 748–751, 872–877, 1596–1607

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

hold and release
BBC 1070
light pens for 690–693
low resolution 26–32
magnification program for 1081–1087
pads 770–774
paged, for animation 1022–1027, 1132–1137
pages, in memory
Dragon, Tandy 215
patterns from nature 1164–1171
perspective drawing 1461–1464
picking and dragging 1000–1004
programming for collisions 656–661
ROM
Commodore 64 420–421
routine for pack of cards 534–540
rubber-banding 998–1000
shading 1464–1465
snow scene
Commodore 64 186–188
solids of rotation 1192–1197
tank UDG 10–15
trace of sound 1092–1095
using fractals 1398–1401, 1434–1439
using GET, GET$ and INKEY$ 132–133
using GET and PUT
Dragon, Tandy 350–352
using LOGO 1296–1300, 1317–1320
using PLOT, DRAW, LINE, CIRCLE and PAINT 84–91
using POKE
Dragon, Tandy 244–245
using SIN and COS 307–308
using STRING$
Acorn, Dragon, Tandy 205–206
using Teletext mode
BBC 28, 1068–1073
wireframe 509–513, 560–565, 605–611
with CAD program 566–572, 573–577
with FORTH 1534
also see animation; movement; ROM graphics; sprite; UDG
Graphics tablet
see joysticks
Graphs
for SIN and COS 254–255
in plant growth program 1052–1054
linear 413–419
negative and positive values in 419
of distribution curves 697–700
of sound envelopes
Commodore 64 1140–1142
scaling factors in 418–419
step 416–417
time distribution 1469
also see bar charts; pie charts
Gravity
effect of when plotting projectiles 743, 802
Gray code 1566–1567
Greensleeves tune,
machine code routine for
Acorn 970–972
Commodore 64 966–970
Dragon, Tandy 972–973
Spectrum 966
Grids
for adventure games 298–299
for perspective drawing 1462
for Teletext graphics
BBC 1068
for UDGs 8–13
in spite generator program
Commodore 64 777–780
in wireframe drawing 512–513, 562–565
in UDG generator program 722–727, 758–764
mapping on graphics pads 772–773
Growth, measuring 1049–1956
Guessing games
see 'match that'; wordgame
Gunsight program 464–468

H

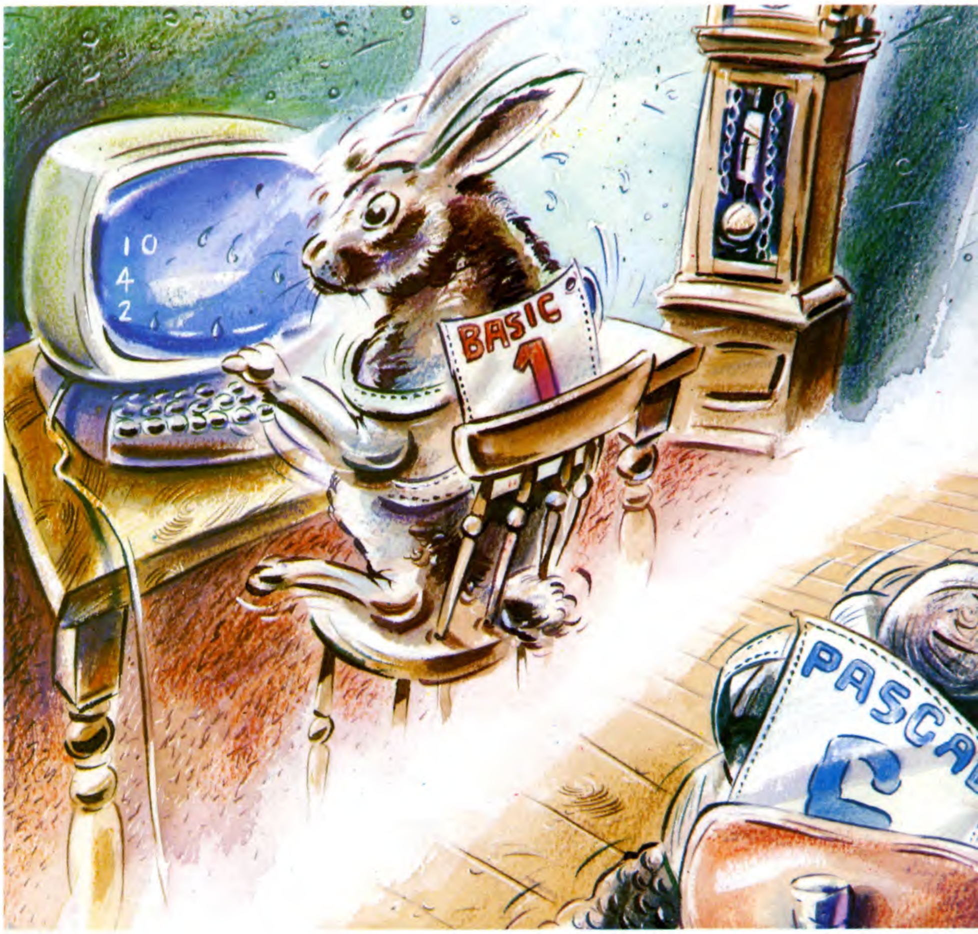
Hacking 612–615
Hand assembling 309–313, 321–327
Hard disks 508
Hardware
see peripherals
Harmonizing in music, programs for
Acorn, Commodore 64 986–991
Hash-coding 1156–1157
Header, in FORTH 1484, 1510
Helicopter graphic
Commodore 64 31
Help routine
in adventure games 425–427
Heuristics, use of in Cavendish Field war game 1373–1377
Hex dump routine, to debug cliffhanger 1580–1588
Hexadecimal
conversion from binary 38–39, 42, 43, 156–160
conversion to decimal
Acorn 157, 593–595
Commodore 64, Spectrum 1527–1530
Dragon, Tandy 157
in screen-scrolling routine 322–327
listing of program to SAVE to disk drive *Vic 20* 682
listing of text-compressor 650–655
listing of trace programs
Commodore 64, Vic 20 514–519
Spectrum 479–483
memory locations 208–215
opcodes listings 311
translation of
see assembly language
use of in machine code 65–66
Hexagon, drawing in LOGO 1296–1300
HICOL *Commodore 64* 87
@HICOL *Commodore 64*
routine to set up 874
High level language
see BASIC; languages, alternative
High-resolution graphics
see graphics, hi-res
HIRES *Commodore 64* 87
@HIRES *Commodore 64* 1596
Histograms
see bar charts
Hobbies file program 46–53, 75–79, 947–952
Home keys 289–290
Horoscope program 1245–1253
House, drawing a
Acorn, Commodore 64 107–109
House purchase, using PERT chart 1430–1433
Housekeeping, definition 1323
Hydraulic ram, program to demonstrate 938–939
Hyperbolas
program to draw 860–861
uses of 863, 894–895



I

Iconic models, definition 1198
Identifier, in Pascal 1386–1387
IF . . . THEN statement 3, 33–36
as program structure 175–178
combination of 35
for graphics effects 366–367
in FORTH 1534
in LOGO 1300
with relational operators 284–288
IF . . . THEN . . . ELSE 3, 37
as program structure 175–178
in FORTH 1534
in Pascal 1388–1390
IF . . . THEN . . . GOTO
Acorn 36
also see while . . . do
Imperative languages 1578
Imperial to metric
conversion program 520–527
Importing data 754
Index
of loop, in FORTH 1534
Indexing
in DBMS 756
of tapes and disks 683
Indirection operators
Acorn 247
Infix notation, in LOGO 1320
INITIALIZE
Commodore 64, Vic 20 822
INK
Spectrum 86
INKEY *Acorn*
use of to detect keypresses 28, 829
INKEY$
Acorn 55, 132–135
Dragon, Spectrum, Tandy 54–55, 103
Input statement
check with ASCII codes 319
in FORTH 1482, 1511
in LOGO 1297
in Pascal 1354–1355
OS routines for 1323
with TAB 118
use of to plot graphs 417
Input buffer *Acorn* 215
Input prompts 130–131
Input/output keywords for datafiles 626–627

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620



Input validation routines 375–379
Insertion sort routine 710–711
INSTR
Acorn, Dragon, Tandy 206, 348
BBC 204
Instructions,
adding to BASIC
Acorn, Dragon, Spectrum 844–851
Commodore 64, Spectrum 1525–1531
in adventure games 424–426
Interfaces
for actuators and sensors 1556
for disk drives 508
for joysticks 223, 224, 468
for printers 229, 644
for *Spectrum* 846
MIDI 1619–1620
people/machine 1554
Interior design with a computer
see room planner program
Interpreted language 1482
Interpreter, definition 921
Interrupt
driven routines
Commodore 64, Vic 20 514–519
Spectrum 478–483
request 1263
use of in background music program
Acorn, Commodore 64 1448–1449
use of in clock routine 896–987
INV *Acorn*
creating a new command 847–849
Invalid values 377
Inventory, in adventure games
267, 364, 387–388, 425, 426
INVERSE *Spectrum*
creating a new command 844–847
Inverse functions, for UDGs 758–764
Inversing the screen
ZX81 432
INVERT *Dragon*
creating a new command 849–851
Irregularities, measuring 1397–1401
Iteration 1458

J

Jeep, drawing a
Dragon, Tandy 91
Joystick 220–224
checking register in cliffhanger
Commodore 64 1341
program for 464–469, 492–497
JOYSTK *Dragon, Tandy* 468–469
Jumps, in assembly language
see addressing
Jungle scene program 485–488, 528–533

K

K, definition of 208
Kempston joystick, program for 464
Kernal ROM
  Commodore 64 212
Key signature, in music 703
Keyboard
  as piano
    see music composer program
  buffer *Acorn* 215
    clearing of 377–378
  control in games 57–59
  matrix of 974–976
  replacement of
    Spectrum 332
  suitability for wordprocessing 544
Keyfield, in DBMS 757
Keywords
  abbreviating 333
    Commodore 64 421
  entering 19
  interpretation of by OS 1323
  tracking down with
    cross-referencer 1512–1519
Keys, function
  Acorn, Commodore 64,Vic 20 825–829
KILL *Dragon* 824
Keypresses
  detecting 54–59
    Acorn, Commodore 64,
      Vic 20 827–829
    by OS 1323
  for direct entry graphics
    BBC 1073
  how they work 826, 974
    in machine code game 929–932
  in quick-draw game 103
  multiple, programming for 974–979
  prompts for 375
  to control background music 1449

L

Labels, in assembly
  language 310, 312, 313
LAMBDA, in LISP 1456
Languages, computer
  alternative 1576–1578
  functional and imperative 1578–1579
  future developments of 1579
  FORTH 1482–1485,
    1508–1511, 1532–1536
  LISP 1410–1415,
    1456–1460, 1578–1579
  LOGO 1264–1268,
    1296–1300, 1317–1321
  Pascal 1352–1355, 1386–1391
    also see assembly language;
      BASIC; machine code
LEFT$ 202–204
Legends, for graphs 416, 419
LEN 204
Letter-generator programs
  811–819, 838–843
Letter-writing
  program for 124–128
  with wordprocessor 542
  with text-editor 852–856,
    878–883, 914–920
Levels of difficulty in games
  in maze game 194–199
  in typing game 355–359
Levers and fulcrums,
  program to demonstrate 933–935
Life
  losing of in games 193–200
Life game 1237–1239
LIFO principle 1484
Light pens 690–693
Light rifle 691
Limit, of loop in FORTH 1534
LINE command *Dragon,*
  Tandy 90–91, 374
    use of to design typeface 840–843
@LINE *Commodore 64*
  routine to set up 876
***LINE** *Acorn* 847–849
Line
  drawing by rubber-banding 998–1000
  drawing in CAD program 566–572
  drawing routine for wireframe
    drawing 511
  numbers in programs 7
    alteration of 340–343
  use of to fill with colour
    Acorn 953–959
Line input
  Acorn, Dragon, Tandy 131
  in FORTH 1484
LISP
  part 1—introduction and
    list-handling 1410–1415
  part II—defining functions 1456–1460
LIST function 4
  to indent
    Acorn 176
  to trace errors 335
  with function keys
    Acorn, Commodore 64,
      Vic 20 827–829
Lists, handling
  with LISP 1412, 1458–1460
  with LOGO 1321
LLIST
  Dragon, Tandy 646
  Spectrum, ZX81 644
Loader program
  Acorn, Dragon, Spectrum,
    Tandy 842–843
LOADing
  compressed escape adventure
    game 1575
  control settings for 22–23
  custom typeface
    Acorn, Dragon, Spectrum,
      Tandy 842–843
  files 1361–1363
  of machine code routines 282
  room plans 1311–1313
  when merging programs 340–343
  with assembler program
    Commodore 64 1214–1215
  with LOGO 1298
  with sprite displays
    Commodore 64 172
Locations, in adventure games
  297–301, 344–349
    altering grid for 423–427
Logic, in basic programming 284–289
Logical expressions
  in LISP 1415
Logical files
  Commodore 64 644, 1360
Logical operators 284–288
  for graphics programming
    Acorn 371–373
LOGO language
  part I—commands and
    Turtle 1264–1268
  part II—Turtle graphics 1296–1300
  part III—sprites, maths and
    word-handling 1317–1321
Loop, in FORTH 1533–1535
LOW COL *Commodore 64* 87

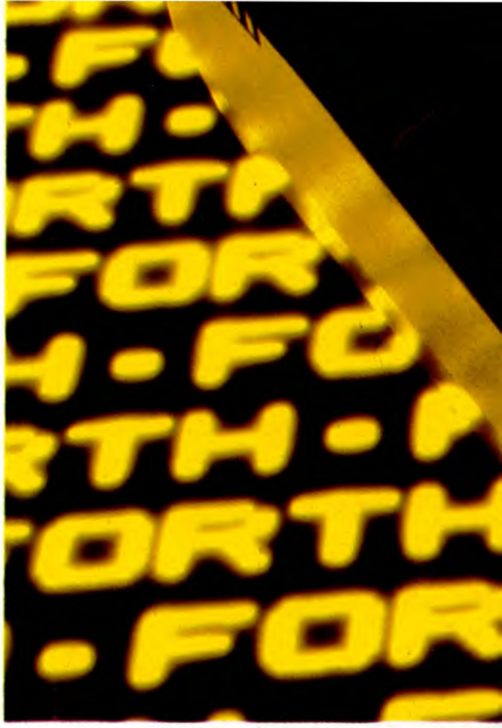
@LOW COL *Commodore 64*
routine to set up 874
Low level language
see machine code
Low-resolution graphics
see graphics, low-resolution
Lower case letters
Dragon, Tandy 142
LPRINT
Spectrum, ZX81 644
Lunar touchdown game 1088–1090

M

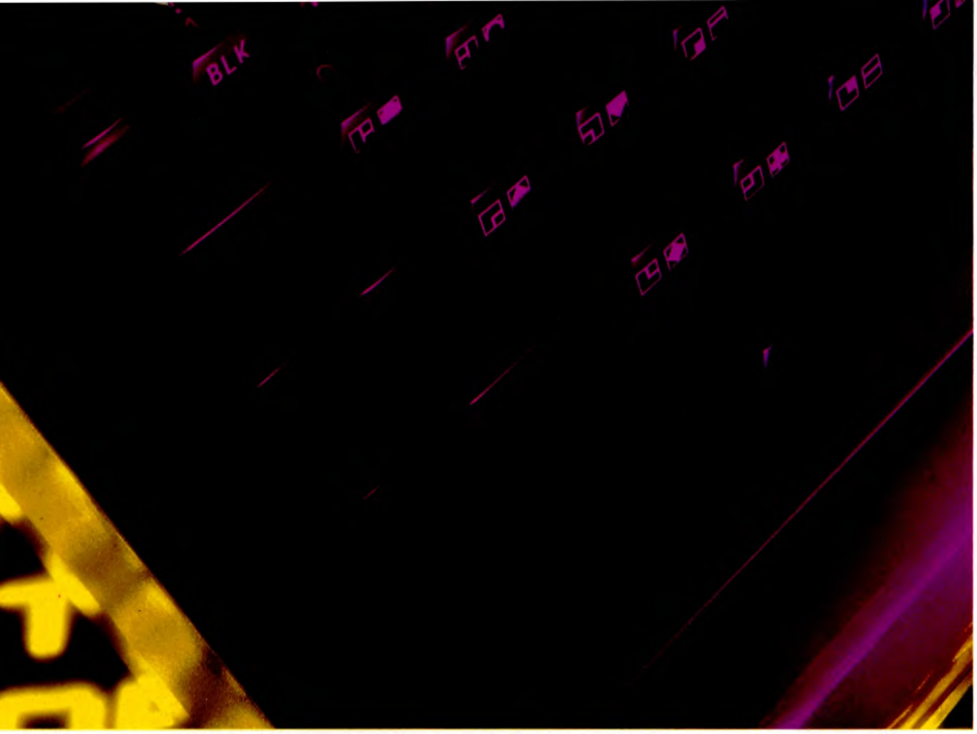
Machine code
advantages of 65–66
assembler program for
Commodore 64 402–405
Spectrum 380–384
converter to SAVE to disk
Commodore 64, Vic 20 676–682
cross-referencer utility 1512–1519
entering programs in 276–283
games programming
see cliffhanger; life game
graphics 38–45, 80–83
routine to add commands for hi-res
Commodore 64 748–751, 872–877, 1596–1607
Vic 20 428–432
listing of UDG frameprint
routine 784–789
LOADing 282
merging routines, problems of 992–997
monitor program 280–281
out command for sound and colour
Spectrum 728–732
routine to alter BASIC
Acorn, Dragon, Spectrum 844–849
routine to play background music
Acorn, Commodore 64 1448–1449
routine to protect programs 462–463
routine to SAVE to Microdrive
Spectrum 616–621
SAVEing with assembler program 1214–1215
speeding up games routines 8–15
stripper routine
Acorn 546–552
Dragon, Tandy 637–641
text-compressor program
coding routine 630–636

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

decoding routine 648–655
timer routine 896–898
tonal screen dump 1369–1371
toolkit, to add to BASIC
Commodore 64, Spectrum 1525–1531
trace program
Commodore 64, Vic 20 515–516
Spectrum 478
translation of 67
tune routine 966–973
Machine stack
definition 237
Acorn 215
Spectrum 211
ZX81 212
Magnification program 1081–1087
Mantissa 792
Maps, drawing in war game 1282–1286
MAPCAR, in LISP 1459
'Match that' colour code
guessing game 1356–1357
Mathematical functions
in fractal geometry 1397–1401, 1434–1439
in mechanics 935
in puzzle-solving 1557–1561
in spreadsheet program 1120, 1184
making pictures with 250–256
speedy use of 923–924
to assess population tendencies 1170–1171
to create bouncing ball graphics 584–592
to draw curves 857–863, 889–895
to draw patterns from orbits 1164–1167
to measure growth 1049–1056
to plot graphs 414–415
to plot projectiles 740–747
use of power function 406–412
with FORTH 1485
with LISP 1415
with LOGO 1320
also see binary; decimal; hexadecimal; number bases; user-defined functions
Matrix generation 974–976
Mazes
in games 68–74
random 193–200
Measurement
conversion program for 520–527
in room planner program 1269–1275
of plant and animal growth 1049–1056
Mechanical advantage,
definition 935
Mechanical devices,
controlling with a micro 1552–1556
Mechanics, programs to show
principles 933–939
Melodies, musical 985–991
Memory
addressing locations 310–313
advantages of Pascal in 1353
area for protection routines 460
banks, range of
Commodore 64 1258–1259
calculating bytes saved by
stripper program 593–595
checking how much
with LOGO 1299
creating space for machine code
programs 276–278
how BASIC programs are
stored in 1106–1112



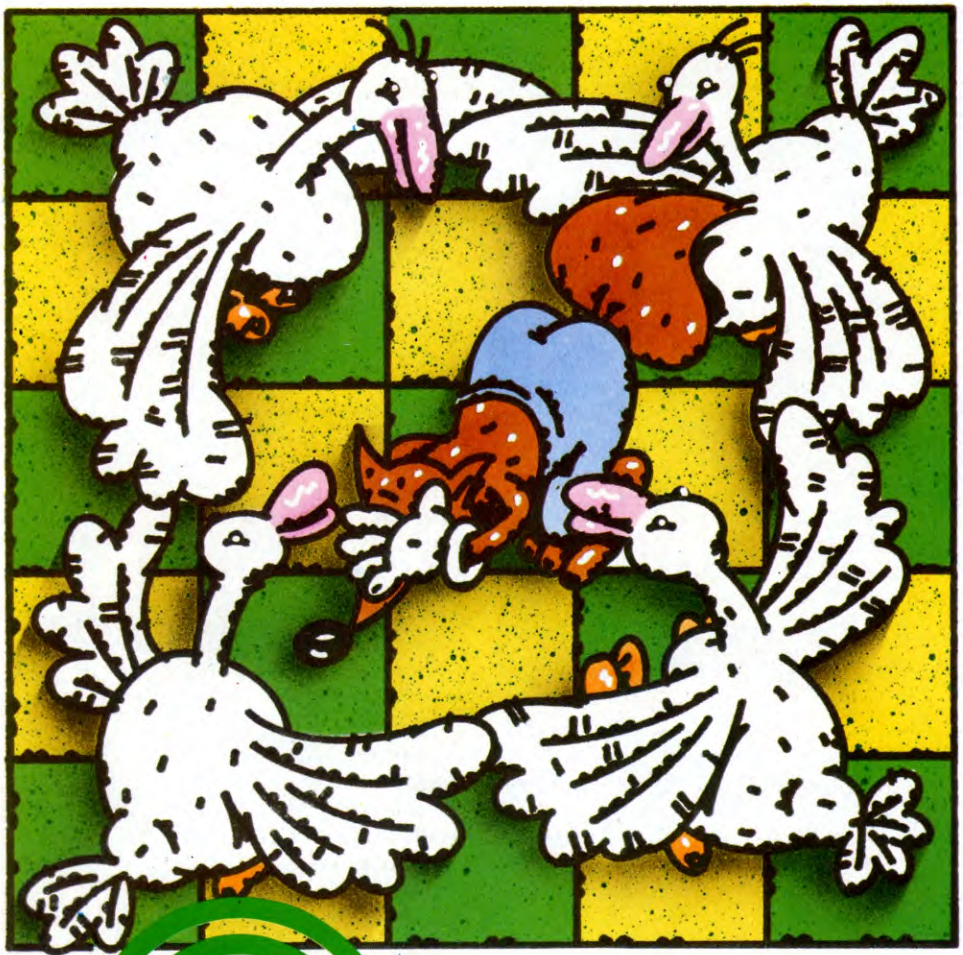
limitations for screen display 448
locations and maps 209–215
locations of VIC-II chip
Commodore 64 1262
looking into, using
PEEK and POKE 240–247
managing by OS 1323–1327
manipulation of
Commodore 64 246–247
mapping, definition 1023
paged graphics in 1023–1027, 1132–1137
requirements of
adventure games 268
Teletext mode
BBC 1068
wordprocessors 544
SAVEing blocks of UDGs
from 532–533, 722–727
saving
versus speed 923
with stripper program
Acorn 546–552
Dragon, Tandy 637–641
with text-compressor 628–636, 684–689
storing new keystrokes in
Acorn, Commodore 64, Vic 20 827–829
storing new typeface in 842
storing sprites in
Commodore 64 1258–1260
storing UDGs in 451–457
use of for sprite control
Commodore 64 171–172
Menu 46
Merging programs 339–343
machine code 992–997
Messages
coded, program for 315–318
encoding and decoding with
text-compressor 684–689
in adventure games 385–386, 628–636
on graphics screen
Dragon, Tandy 191–192



Metric to imperial
conversion program 520–527
Microdrive *Spectrum*
connecting and operating 820–822
conversion option for hobbies file 947
input/output keywords for 626
map areas 209
SAVEing to 616–621
storing datefiles on 1358–1360
Microprocessor, how it works 236–239
MIDI interfaces 990, 1619–1620
MID$ 202–204
Minefield game 97–103
Mirror function, for UDG 758–764
Missiles, firing 55–58
in Cavendish Field 1346–1348
Mnemonics 309–313
in assembler program
Commodore 64 404
Spectrum 383–384
MOB
Commodore 64 15
MOD *Acorn* 205
Modems 615, 712–715
MODE *Acorn* 28, 88–89
designing dumps for different 1368–1369
in random maze game 197
memory locations 214
with POKE
BBC 241
Modelling reality 1198–1204
in food stall program 1209–1213
Modules
see subroutines
Monitors and TVs 445–449
checking with test card program 1474–1475
Monkey UDG
see jungle scene
Morse code program 963–965
Motorbike UDG *Vic 20* 428–429
Mountain, program to model 1436–1437
Mouse
see joysticks
Movement
of furniture in room planner 1308–1313
of projectiles 740–747
of ROM graphics 26–32
of sprites
Commodore 64 1262–1263
of troops in war game 1288, 1305–1306
of UDGs 8–15, 57–59, 81–83, 98–99, 146–150, 428–432
routine in adventure game 348–349
through mazes 68–74, 194–200
also see animation
MSX standard BASIC 1576
MULTI *Commodore 64* 87, 186–188
@MULTI *Commodore 64*
routine to set up 872–874
Multi-key control, programming for 974–979
Multicolour
background graphic
see jungle scene
sprites
Commodore 64 776–783
Multiple search option, in hobbies file
Acorn 950–951
Commodore 64 948
Dragon, Tandy 952
Multiple statement, in Pascal 1354
Multiplication
code program 1046–1947
on eight-bit processor 1146–1147
tables, program for 5–7
Music
analyzing and storing 1091–1095
background, program to play
Acorn, Commodore 64 1448–1449
chords and harmonies 985–991
composer program
part 1 1333–1337
part 2 1392–1396
part 3 1416–1423
converting keyboard to play 672–675
data compression for 1222–1229
frequency in 671
machine code routine for 966–973
notation of 701–702
ruler for pitch values 702
simple tunes 675
tempo 704
theory of scales 670–671
transcribing for computer 704–707
twelve-bar blues program 1222–1229
using **out** *Spectrum* 728–732
with synthesizers, using MIDI 1616–1620
also see sounds

N

Negative numbers
binary, conversion program for 180–183
in drawing *Spectrum* 85, 108
storing in memory 793
with SQR function 412
Nested structures
IF . . . THEN statements 175–176
loops 19
in FORTH 1534–1535
Networks 614, 715
PERT 1466
NEW 9–15
Nibble 238
NIL, in LISP 1415
90% sure time 1467
Nodes, memory in LOGO 1299
Noise sound *Commodore 64* 232
Noises
see sound effects
Nonary numbers 111
NOP machine code instruction 908
Normal Gaussian curve 1467
NOT 286–288
with PUT in graphics programming
Dragon, Tandy 352
@NRM *Commodore 64*
routine to set up 872
NULL, in LISP 1415
Null strings 96, 205
NUMBER, in FORTH 1511
Numbering records, for filing 47
Numbers
bases, conversion program for 110–116
decrementing in snake game 804–810
Fibonacci 1056
generation program 1054–1055
handling with LISP 1414–1415
how computers store 790–796
Numeric arrays, in games 155
Numeric variables
see variables
Numerical sorts 392–397
Nybble
see nibble



O

Object list, in LISP 1415
Objects, in adventure games 267, 297, 360–365, 387–388
choosing your own 424, 427
Off-screen drawing 565
Offsets
see registers
OLD *Dragon*
creating a new command 849–851
ON . . . GOSUB 64
for multiple choices 177
Opcodes 67, 309–313
OPEN
Acorn 1361
Commodore 64 135, 626–627, 644, 1360–1361
Operands, in FORTH 1532

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

Operating system
accessing 1324–1327
how it works 1322–1324
software
Acorn, Commodore 64, Vic 20 826–828
Vic 20 214
Operating system for disk drives 508
Operators
see logical and relational operators
Options, in two-dimensional arrays 272–275
OR
in FORTH 1533
in graphics programming
Acorn 371–373
Spectrum 367
with PUT in graphics programming
Dragon, Tandy 352
Orbits
of projectiles 800–802
patterns from 1164–1171
OS command line interpreter (OSCLI)
Acorn 1324–1326
OSBYTE *Acorn* 1324–1326
OSWORD *Acorn* 956, 1326
Othello board game
part 1 980–984
part 2 1005–1009
OUT command, machine code
for sound and colour
Spectrum 728–732
Out of range line numbers 63–64
Output
buffer
Acorn 215
in FORTH 1510–1511
Overwriting
files on disks
Commodore 64, Vic 20 676
how to avoid when merging 992–997
in tonal screen dump 1369–1371

P

Pads, graphics 770–774
PAGE *Acorn* 247, 1109
Page, of memory space 209
Paged graphics, for animation 1023–1027, 1132–1137
PAINT
Dragon, Tandy 91
Vic 20 88
@PAINT *Commodore 64* 1596
Painting by numbers 18
Palindrome solution program,
in Pascal 1388–1390
PAPER *Spectrum* 86
Paper, for printers 228–229
Paper, scissors, stone
games 1502–1507
Parabolas
drawing in graphs 415
in swimmer program 891–893
of orbits 1164–1166
program to draw 859–860
simulating in projectile routine 744–747
uses of 863
Parameters 64
for hi-res graphics
Commodore 64 872, 1596–1599
for user-defined functions 580
input/output 216
passing by recursion 1289
Party game
see horoscope program
Pascal
part 1—algorithms 1352–1355
part 2—commands 1386–1391
Pascal's Triangle 697
Password
program 133–134
use of with bulletin boards 715
Patterns
drawing in LOGO 1296–1300
flashing 366–374
from nature and orbits 1164–1171
of uni-cellular organisms
in life game 1237–1239
patchwork 18–21
using curves 894–895
PAUSE command
Commodore 64 87, 896
Spectrum 896

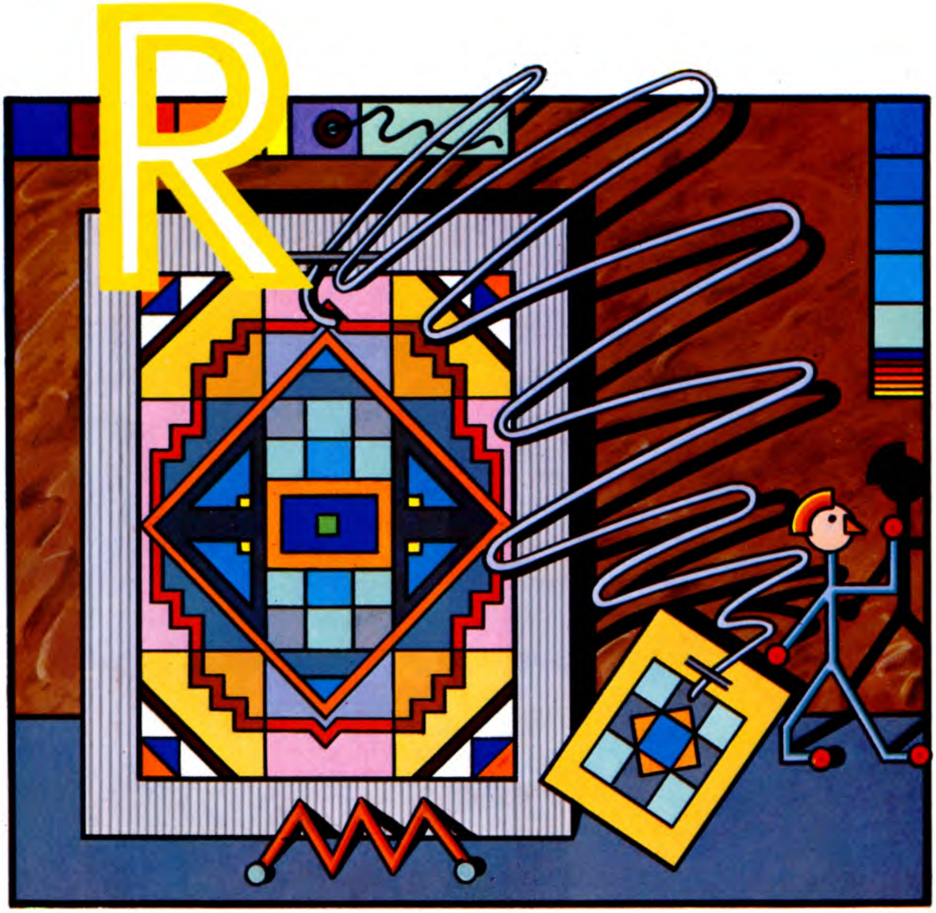
Pause
facility
Commodore 64 1214–1215
Spectrum 482
loop
for musical pitch
Spectrum 728–732
in machine code game
Dragon, Tandy 912–913
Spectrum 906–907
PCLEAR *Dragon, Tandy* 278
PCLS *Dragon, Tandy* 374
PCOPY *Dragon, Tandy*
for animation 1027, 1137
in flight simulator 720, 739
PEEK
at BASIC programs in memory 1105–1112
for merging programs
Dragon, Tandy, Vic 20 341–343
how it works 211–212, 240–247
use of in collision detection
Commodore 64, Vic 20 658–659
Peripherals
control of by micro 1552–1556
controlling a stepper motor
BBC 1566–1567
MIDI systems, for music 1616–1620
robotics 884–888
also see disk storage; graphics pads; light pens; printers; tape storage
Perspective drawing 1461–1465
in wireframe drawing 605–611
PERT program
part 1—the database 1429–1433
part 2—using the program 1466–1473
Pets survey program 270–275
Picking and dragging 1000–1004
Pie charts 474–476
Pitch, musical 669–675
analyzing for data compression 1224
efficient use of
Acorn 1226
ruler for 702
setting with assembly language routine
Spectrum 728–732
Planets
program to draw 1465
program to show motion of 802–803
Plant growth program 1052–1053
PLAY *Dragon, Tandy* 73, 234–235, 674
PLIST, in LISP 1460
PLOT 84–91
in graph program 254–255
new commands
Acorn 953–959
use of to design typeface
Acorn, Spectrum 838–843
@PLOT *Commodore 64*
routine to set up 874–876
PMODEs *Dragon, Tandy* 12, 40, 90–91, 248–249
memory locations 215
with GET in graphics programming 351
Pointer, in memory 209
adjustment for program protection 461
resetting for UDG bank
Acorn, Commodore 64, Spectrum 451–457
resetting with OLD
Dragon 849–851
sprite
Commodore 64 172, 1260–1261
Poisson Processes, definition and program to demonstrate 1199
POKE
for better screen displays
Commodore 64 435
for merging programs 341–343
how to use 240–247
in music programming
Commodore 64, Vic 20 672–673
to access OS
Spectrum 1324
to clear memory space
Commodore 64 278, 404
to create sound effects
Commodore 64, Vic 20 232–233
to design new typeface
Commodore 64 839–842
to disable keys
Commodore 64, Dragon, Tandy, Vic 20 379
to enable and store sprites

Commodore 64 1259–1263
with GET and PUT
Dragon, Tandy 351–352
Polygons, program to draw 893–894
Pontoon game
part 1—graphics and shuffling 534–540
part 2—player's turn 553–559
part 3—computer's turn 598–604
Pools simulation program 1158–1160
Ports, input/output 728
addresses for keyboard
Spectrum 974
number of
Spectrum 730
POS function
Commodore 64, Vic 20 122
Postbyte
see addressing
Postfix notation 1485
Power function, uses of 406–412
Powering up, disk drive 820–822
PPOINT *Dragon, Tandy* 661
Predicates, in LISP 1415
Prediction by computer 694–700, 1158–1163, 1198–1203, 1209–1213
in horoscope program 1245–1253
Prefix notation
in LISP 1415
in LOGO 1320
PRESET
Dragon, Tandy 90, 352, 374
Primitives, in LOGO 1267
PRINT command
during trace program
Spectrum 478
for better screen display 117–123
in LISP 1415
in LOGO 1320
to create ROM graphics 26–32
PRINT AT *Spectrum, ZX81*
for better screen displays 120, 434–436
for movement 32
with SIN and COS 303–305
PRINT@ *Dragon, Tandy* 121–122, 435–436
PRINT SPC
Commodore 64, Vic 20 434–435
PRINT TAB
for better screen display 117–122, 434–438
for movement
Commodore 64, Vic 20 30
for UDG grid
Acorn 11
PRINT USING *Dragon, Tandy* 796
Printer
best type for wordprocessing 545
buying a 225–229
connecting up 643–644
control commands for 644–646
dip switches in 646
escape codes for 646–647
paper for 643
ribbons for 643
screen dump programs for 1365–1371
self-test on 643
use of for letters 124–125
use of with text-editor 914–920
Printer buffer
Acorn 215
Spectrum 209
Priority, of sprites in memory
Commodore 64 783

Probability, computing
see prediction by computer
PROCedures *Acorn*
advantages of 922, 924
calling by recursion 1289–1295
in bubble sort routine 217
in random maze game 197–198
use of to fill with colour 954–959
Procedures, in LOGO 1268
drawing patterns with 1296–1300
erasing, LOADing and SAVEing 1298–1299
listing from memory 1299
Program counter
see registers
Program structuring
see structured programming
Program symbols, chart of
Commodore 64 421
Project planning, using
PERT 1429–1433, 1466–1473
Projectiles, plotting 740–747
in space 800–802
over barriers 797–800
Prompts
best use of 375–376
in adventure games 347
Properties, in LISP 1460
Protecting your programs 459–463
PSET *Dragon, Tandy* 90, 145, 352
for colour effects 374
to draw graphs 255
Pulleys
program to demonstrate 935–938
Pulse wave
Commodore 64 1417
Punctuation 4
errors in 36, 337
for better screen display 436–439
in DATA statements 106
in file names 24
in FORTH 1484–1485, 1510–1511
in LISP 1412
in LOGO 1320–1321
in Pascal 1354–1355, 1391
in PRINT statements 119–123
when handling files 1360–1363
PUT
in graphics programming
Dragon, Tandy 41, 167, 350–352
Puzzles, solving on your
computer 1557–1561

Q

Quicksort routine 711
recursive 1293–1294
QUOTE, in LISP 1412
Quote mode *Commodore 64* 420



Radians conversion program
Commodore 64, Dragon, Spectrum, Tandy, Vic 20 250–251
Radio codes, deciphering 1562–1565
RAM, definition 209
character set
Commodore 64 44
program to PEEK into 242–243
pointer for program protection 461
RAMTOP 211–212
lowering of
Dragon, Tandy, Vic 20 278
Spectrum, ZX81 276
RAND USR *ZX81* 432
Random access file 622
Random maze game 193–200
adding sound effects to 231–235
Random numbers
generating 2–7, 1160–1161
in samples and surveys 1161–1163
also see RND function
Random variables
normal, in food stall model 1209–1213
types and distribution of 1198–1203
RANDOMIZE 2
Raster interrupts
Commodore 64 782,1263
Raster scan 447
READ 104–109
REC *Commodore 64* 87
@REC *Commodore 64*
routine to set up 876–877
Record, in datafile 624
Rectangle option, in
CAD program 573–577
Rectangles, program to draw 1056
Recursion
in BASIC 1289–1295
in fractal programs 1398–1401, 1434–1439
in LISP 1458–1459
in LOGO 1299–1300
also see quicksort
Reducing images
see magnification program
Refresh register *Spectrum* 1441
Registers, definition 236
addressing 310–313
names of
Acorn, Commodore 64, Vic 20 238
Dragon, Tandy 239
Spectrum, ZX81 238
VIC-II *Commodore 64* 781–783
Regulatory control systems 1552–1553
Relational operators 35, 284–288
Relative addressing
see addressing
Relative copying, in spreadsheet
program 1184
REM statements 63, 176
for machine code programs
ZX81 276
stripper routine to remove
Acorn 546–552
Dragon, Tandy 637–641

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

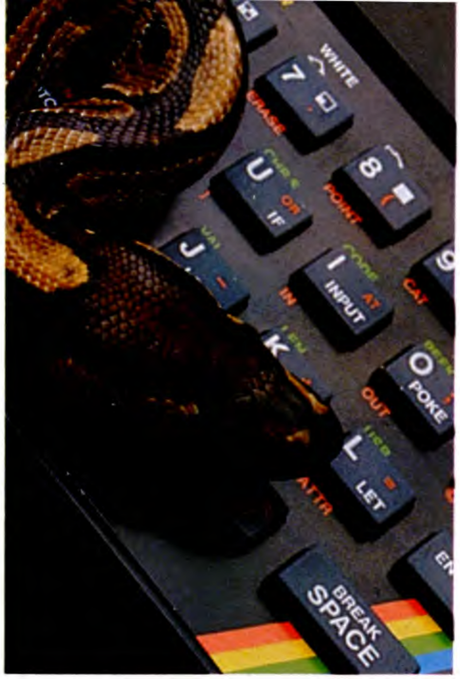
- **REMPROP,** in LISP 1460
- **RENAME** *Acorn, Dragon* 828
- **RENUMBER** 342
- **Renumbering,** machine code routine for *Commodore 64, Spectrum* 1525–1530
- **REPEAT,** in LOGO 1268
- **REPEAT . . . UNTIL** 18, 36, 177–178, 582
  - in Pascal 1387–1388
- **Repetition techniques,** in Pascal 1387–1390
- **Replace option,** in wordprocessing 543
- **Report,** in DBMS 757
- **Resonance,** program to simulate 1169
- **Resolution,** checking with test card program 1474–1475
- **Responses,** in adventure games 344–348
- **RESTORE,** command and pointers 106–107
- **RETURN** 62
  - in accounts program 136–137
  - use of when merging
    - *Acorn* 342
  - use of with function keys
    - *Acorn* 828–829
    - *Commodore 64, Vic 20* 827–828
- **Reverse graphics symbols**
  - *Commodore 64* 420
- **Reverse Polish Notation** (RPN) 1485
- **RIGHT$** 202–204
- **RND function** 2–7
  - for graphics effects
    - *Acorn* 189
  - in anagram game 202–204
  - in bluffing games 1500–1507
  - in Cavendish Field 1282–1288, 1307
  - in coin and dice games 61–63
  - in flight simulator 733–739
  - in food stall model 1209–1213
  - in pontoon game 600–604
  - in pools simulation program 1158–1160
  - in sample and surveys 1161–1163
  - to draw spiral patterns 308
  - to model asymmetry 1436–1437
  - to transform distribution of variables 1202
- **Roadway perspective program** 1132–1134
- **ROBOL language** 887
- **Robotics** 884–888, 1266
  - first principles of 1566–1567
- **ROM,** definition 209
  - changing for LISP 1412
  - graphics
    - *Commodore 64* 226, 420–421
    - enlarging
      - *Acorn, Commodore 64, Spectrum, Vic 20* 811–815
    - simple animation of 26–32
    - to create maze games 68–74
  - program to PEEK into 242
  - switching between old and new
    - *Spectrum* 846–847
- **Room planner program**
  - part 1 1269–1275
  - part 2 1308–1313
- **ROT** *Commodore 64* 188
- **@ROT** *Commodore 64* 1598
- **Rotating**
  - bits 1038–1039, 1147, 1217, 1244
  - conic sections 861–863
  - furniture in room planner 1308–1313
  - solid drawings 1192–1197
  - UDGs 758–764
- **Routine table,** for hi-res graphics
  - *Commodore 64* 877
- **RTTY codes** 1562
- **Rubber-banding** 998–1000
  - for solids of rotation 1192–1197
- **RVS** *Commodore 64* 31

S

- **SAM chip** *Dragon, Tandy* 1043
- **Samples and surveys** 1161–1163
- **Satellites,** tuning in to 1564–1565
- **SAVEing**
  - control settings for 22–23
  - custom typeface 842–843
  - designs from graphics pads 771, 774
  - files 48, 77, 1358–1361
  - machine code routines 281
  - option for on assemblers
    - *Commodore 64* 404, 1214–1215
    - *Spectrum* 384
  - room plans 1311–1313
  - sprite displays
    - *Commodore 64* 172
  - to disk drive
    - *Commodore 64, Vic 20* 676–682, 824
    - *Dragon, Tandy* 824
  - to Microdrive
    - *Spectrum* 616–621, 822
  - verifying 24
  - when merging 340–343, 992–997
  - with LOGO 1298
- **Saving memory space** 333
- **Sawtooth sound** *Commodore 64* 232
- **SC (selected character)** *Commodore 64* 44
- **Scales,** musical 985
- **Scaling**
  - custom typeface 841–843
  - graphs 418
  - in magnification program 1081–1087
  - parabolas and hyperbolas 859–861, 863
- **Scatter sort routine** 710
- **Scoring,** in games 68–74, 100–103
  - in cliffhanger 1101–1105, 1145–1151, 1404–1409, 1440–1447, 1476–1481
    - also see life, losing a
- **Scrambling routine,**
  - in anagram game 202–204
- **SCREEN** *Dragon, Tandy* 40, 90–91, 248–249, 439
  - for flashing 166
- **Screen display**
  - as two "windows" 1257
  - dumping of 1365–1371
  - formatting 117–123
  - improving 433–439, 581–582, 811–819
  - SAVEing and LOADing
    - *Acorn, Dragon, Spectrum, Tandy* 842–843
  - use of several in cliffhanger 904
- **Screen dump**
  - for graphics pads 771
  - from UDG bank 759–764
  - of graphics 1365–1371
  - with COPY
    - *Spectrum, ZX81* 644
- **Screen memory** *Acorn* 214
- **Screens,** in FORTH 1482, 1510
- **Scrolling of screen**
  - backwards 282
  - in cliffhanger 1034–1043, 1145–1151
  - sideways 323–327
- **Search routines**
  - binary and serial 924–927
  - in text-editor program 914–920
  - single pass 1162–1163
- **Searches,** in DBMS 755–757
- **Secondary address,** of datafile
  - *Commodore 64* 1360
- **Sectors,** on disks 1608
- **Seikosha printer codes** 647
- **Self-similarity,** definition 1398
- **Sensors** 1554–1555
  - on light pens 691
- **Squencer,** music 1620
- **Sequential file** 622, 753–754
- **Serial access,**
  - from storage device 505–506
- **Serial file** 622
- **Serial search routine** 924–925
- **Servos,** use of on Beasty 887–888
- **SET,** in LISP 1413
- **SGN** 150
- **Shading effects,** with colour 1464–1465
- **Shading the screen**
  - *Spectrum* 85, 366–367
- **Shapes**
  - detection of in screen display 656
  - generator program
    - *Acorn, Dragon, Spectrum, Tandy* 1437–1439
  - manipulating
    - see picking and dragging
- **Sharing information**
  - see communications
- **Sharps and flats,** in music
  - definition 670–671
  - transcribing 702–703
- **Shell,** firing a 10–15
- **Shell-Metzner sort routine** 397
- **Shell sort routine** 396–397
- **Shields,** use of in games 144–151
- **Short-wave radios,**
  - different types 1562
- **Shortening** programs 333
- **SID chip** *Commodore 64* 231, 968
  - in music programming 986–991
- **Sideways scrolling routine,**
  - in assembly language 323–327
- **Sign convention** 183
- **Simons' BASIC** *Commodore 64* 87
  - graphics commands in 368–370
- **Simulation**
  - see modelling reality
- **SIN** 250–256, 302–308
  - in fractal programs 1398–1401
- **Sine wave,** in graph 414–415
- **Singer animation program**

Acorn 1026–1027
Single pass search 1162–1163
Single Side Band (SSB)
radio signals 1562
Siren sound effect
Acorn 234
Dragon, Tandy 235
Slipping ladder program 890–891
Smalltalk 1579
Smart software 714
Snake game 804–810
Snake UDG
see jungle scene
Snap 888
Snooker ball program 584–592
Snow scene *Commodore 64* 186–188
Snowflake, program to draw 1434–1436
Snowflake curve 1398
Solenoids 1555
Solids of rotation 1192–1197
Sort routines 216–219, 392–397, 708–711, 914–920, 947–952
also see quicksort
SOUND
Acorn 233, 674, 1138
Dragon, Tandy 234
Sounds
analyzing and sorting 1091–1095
envelopes for modifying
Acorn, Commodore 64 986–991, 1138–1144
facilities for 1616
in cliffhanger 1242–1244, 1440–1447
using machine code **out**
Spectrum 728–732
using PEEK and POKE
Spectrum 243
also see music
Sound workspace *Acorn* 215
Space station
drawing a wireframe 666–668
game 144–151
Spaces
as cause of error 337
for screen formatting
Commodore 64, Vic 20 122
insertion of in text 883
missing out 333
stripper routine to remove
Acorn 546–552
Dragon, Tandy 637–641
use of in FORTH 1484, 1510
use of in wordgame 899–903
SPC function *Commodore 64* 122
Speech synthesizers 398–401
Speed game typing exercise 355–359
Speed POKE *Dragon, Tandy* 444
Speeding up
BASIC programs 333, 921–927
also see stripper programs
cliffhanger 1476–1481
games using PEEK and POKE
Dragon, Tandy 245–246
sort routines 395

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620



Spelling, help with by
workprocessor 543–544
Spiral, drawing a 307–308
with LOGO 1300
Spreadsheet program
part 1 1118–1126
part 2 1172–1176
part 3 1184–1191
Sprites *Commodore 64*
defining and SAVEing 168–172
explosions 164
fire-breathing dragon 83
generator program 776–779
pointers 172
registers to control 781–783
SAVEing 780
setting colour of 1102–1103
storing 779–780
and moving 1258–1263
Sprites, LOGO 1317
animation of 1319–1320
editing 1318–1319
Squares and cubes
see power function
Square comparison program 407–410
Square roots
see SQR function
SQR function 60–61, 410–412
St. Cyr cipher program 962–963
Stack pointer 237
Stack manipulation, in FORTH 1484–1485
Star animation program
Dragon, Tandy 1027
Standard deviation in probability
theory 700
Statistics, use of in
bluffing games 1501–1507
also see samples and surveys
STATUS *Commodore 64* 1363–1364
STEP 17–18
in 24-hour clock program 302–306
to draw spirals 307–308
to plot circles 256
use of for endless loop
Spectrum 581–582
Stepper motors 1555
controlling with BBC micro 1566–1567

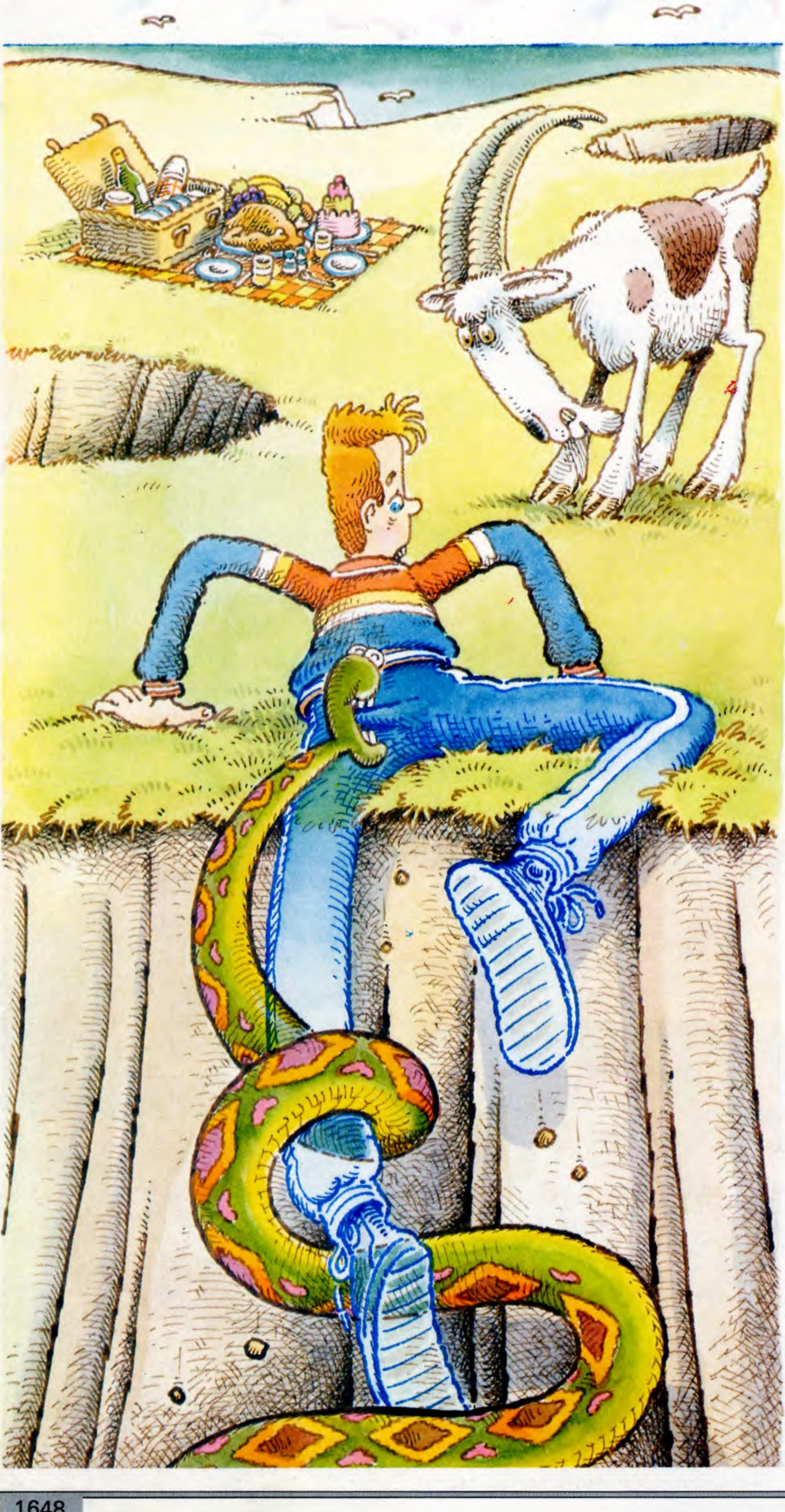
Stepwise refinement 1354
Stochastic models 1198
STOP *Spectrum, ZX81* 64
Stop-frame animation 1022
Storing information
see arrays; files
Storing large numbers 160, 211, 237
Strategy in games
see Cavendish Field
String
arrays, in games 155
comparisons, using ASCII codes 319
functions, user-defined 581–582
in custom typeface program 841
slicing 102, 202–204
storing in memory 1108–1112
tracking down and replacing with
cross-referencer utility 1512–1519
use of for Teletext graphics
BBC 1068–1073
use of with function keys
Acorn, Commodore 64 826–828
also see variables
STRING$ *Acorn, Dragon, Tandy* 98, 205–206
Stripper program
Acorn 546–552
Dragon, Tandy 637–641
routine to calculate bytes saved by
Acorn 593–595
STR$, in clock program
Dragon, Tandy 305–306
Structured programming
definition and flow charts 173–178
with FORTH 1482, 1532–1536
with LISP 1458–1460
with Pascal 1352–1355, 1386–1391
Structures, improving BASIC 922
Stubs *Dragon* 849–850
Stunt-rider UDG *Vic 20* 429
Stylus, for graphics pads 773
Submarine UDG *Vic 20* 430
Subroutines 62–64
calling by recursion 1289–1295
handling of by stack 238
merging 339–343
testing of 218–219
typing in 77
use of for speed 922
Subset, in FORTH 1484
Swimmer in river program 891–893
Symbolic models 1198
Symbols
arithmetic 6
keyboard
Commodore 64 420–421
musical 702
to format letters 125
SYNC *Dragon, Tandy* 1244
Synthesized sound effects,
in music composer program
Commodore 64 1416–1418
Synthesizers, musical 1616–1618
SYS *Commodore 64* 462
to access OS 1324
System input/output and control interfaces
Vic 20 214
System variables
area 209–215
updating for Microdrive
Spectrum 616–621

T

T, in LISP 1415
Tables, multiplication, program for 5–7
Tablets, graphics 770–774
Tabulation
see PRINT TAB; formatting
Tail recusion, in LISP 1459
TAN 250–256
Tank UDG, creating and controlling 10–15
Tape catalogue, new command for
Spectrum 1525–1528
Tape loops 505
Tapes
choice and storage of 23–25
protection of 683
storage on 504–505
also see files
Telephone
directory program 105
use of for hacking 612–615
Teletext mode *BBC* 28, 1068–1073
Teletext services 614, 715
Tempo, altering in music
programming 1229
Terminating numbers 34
Test card program 1474–1475
Test loop
in Cavendish Field 1288
TEXT *Commodore 64* 369
@TEXT *Commodore 64* 1598–1599
Text compressor program
part 1—coding routine 628–636
part 2—decoding routine and
hex listing 648–655
part 3—how to use it 684–689
Text editor program
part 1—basic routines 852–856
part 2—editing facitities 878–883
part 3—sorting, searching,
formatting and printout 914–920
also see editing; wordprocessing
Text screen
memory area
Dragon, Tandy 215
setting up in Cavendish Field 1257
Three Blind Mice tune,
program to play 704–707
Acorn, Commodore 64 990–991
Three-dimensional
bar chart 473–474
drawing 1192–1197
also see perspective drawing
wireframe drawing 509–513, 560–565
TIME *Acorn* 896
Time distribution graphs, for PERT 1469
Time signature, in music 704
TIMER *Dragon, Tandy* 896
Timer routine
for BASIC lines 922
machine code 896–898
Spectrum 1324
Timing, in games
see scoring
Timing and sequence control systems 1552
Title page
formatting a 433–439
machine code routine for 906–913
using Teletext mode
BBC 1071–1072
with a new typeface 811–819
Tokens, in memory 210–215, 1106–1112
Commodore 64 421
recognizing new 844–851
Tones and semitones, in music 671
Toolkit, to add commands to BASIC
Commodore 64, Spectrum 1525–1531
TOR Teletype radio signal 1564
Torus, drawing a wire frame 666
Tower of Hanoi program 1294–1295
Trace program
Commodore 64 514–519
Spectrum 477–483
Trace, sound 1092–1095
Tracker ball
see joysticks
Tracks, on disks
Acorn 1612–1613
Commodore 64 1608–1610
Dragon 1614
Trajectories, plotting 740–747
in space 800–802
over barriers 797–800
also see orbits, patterns from
Trampolinist animation program
Spectrum 1025–1026
Transforming coordinates, in wireframe
drawing 561–565, 605–611
Transmission rate, from tapes 505
Transportability, of programs 1482
Tree search, for evaluating
moves in board games 1098–1099
Triangle sound
Commodore 64 232
Triangles, use of to model
a mountain 1436–1437
TROFF *Dragon, Tandy* 444
TRON *Dragon, Tandy* 444
Truth tables 286
Tunes
playing by ear 671–672
program to compose 1333–1337, 1392–1396
simple, for musical keyboard 674–675
Turtle 885–888, 1266–1268, 1296–1300
TV commands
BBC 189
Twelve-bar blues program 1222–1229
Twos complement 179–183
Typeface, setting up new 811–819, 838–843
Typing course
part 1—home keys 289–295
part 2—QWERTY and alphabet 328–332
part 3—character keys and
improving your speed 353–359
part 4—typing long passages 498–503
Typing, speeding up using function keys
Acorn, Commodore 64, Vic 20 825–829

U

UART chip 229
UCSD Pascal 1390
UDGs
advantages of 484
alien
ZX81 430–432
animation of with GET and PUT
Dragon, Tandy 350–352
bicycle
Dragon, Tandy 350–351
colour
Dragon, Tandy 248–249
creating 38–45, 451–457
disassembling grid routine for 784–789



fire-breathing dragon 81–83
frog 8–15
generator program 721–727, 758–764
grid for 8–13
in Cavendish Field 1254–1256
in jungle scene 484–491, 528–532
SAVEing to tape 532–533, 722–727
storing 532–533, 721–727, 758–764, 1028–1033
stunt-rider
Vic 20 428–430
submarine
Vic 20 428–430
to create block letters
Acorn 815–819
to enlarge ROM characters
Acorn, Spectrum 811–815
Uploading 614, 712–715
User-defined functions
Acorn, Commodore, Dragon, Spectrum, Tandy, Vic 578–583
in FORTH 1484
in LISP 1456–1459
key for
Acorn 570–572
User memory
Vic 20 213
USR function
Commodore 64, Dragon, Tandy, Spectrum, ZX81 282–283
to access OS
Acorn 1324–1326
Spectrum 1324
Utility packages *Commodore 64* 827

V

VAL 101, 205, 795
Values, in LISP 1412
Vanishing point, in perspective 1461
Variables 3–5, 92–96
checking for error 336
comparison of 201–202, 284–285, 319
control 94, 178
dummy 580
during recursion 1291–1293
for sound effects
Dragon, Tandy 235
global, *Acorn* 217
in bubble sort routine 216–219
in FORTH 1508–1510

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

in Pascal 1354–1355
keeping value of 100–101, 216
list of for adventure game 425–427
local
Acorn 217
managing for program speed 923–925
naming of 17, 95, 333
numeric 92
replacing with cross-referencer 1512–1519
setting in cliffhanger 1127–1131
storing in memory 1106–1112
string 4–5, 94–96, 201–207
slicing and sorting 202–204
to move clock hands
Acorn, Dragon, Tandy 303–306
use of with READ and DATA 104–109
Variables area *Spectrum, ZX81* 210–212
VDG chip *Dragon, Tandy* 1043
VDU command *Acorn* 29, 42–43, 147–149, 320, 646
for sideways scroll 327
to turn off cursor 99
Vectors
Commodore 64 460
redirecting 844–851
Velocity, in collision detection routine 656–661
Verbs, as instructions in games 349, 363, 425, 426
VERIFY 24–25
VIC-II chip *Commodore 64* 1258
memory locations 172, 1262
use of to control sprites 781–783
Vic Super Expander for *Vic 20* 88
Video generator chip
Dragon, Tandy 215
Videotex 614, 715
Viewdata 614, 715
Virtual memory 544
Vocabularies, in FORTH 1484, 1510–1511
Volatile storage 504
Volume settings
for sound effects
Commodore 64 232
for sound trace 1092
Volume versus surface area, program to measure 1050–1051

W

War games
see Cavendish Field game
Water pump animation program
Dragon, Tandy 1137
Waveforms *Commodore 64*
displaying and storing 1092–1095
for sound effects 232
in music programming 986, 1417–1418
modulation of 1138–1139, 1142
When the Saints Go Marching In program
Acorn, Commodore 64 986–989
WHILE . . . DO 176–177
in Pascal 1387–1390
White noise 1138
Winchesters
see hard disks
Wireframe drawing
part 1—basic routines 509–513
part 2—three-dimensions 560–565
part 3—different viewpoints and perspective 605–611
part 4—multiple cubes, globe and torus 662–668
solids of rotation 1190–1197
WORD, in FORTH 1511
Word-handling, with LOGO 1320–1321
Word wrap 545
Wordgame
part 1—basic routines 899–903
par 2—adding the options 940–945
Wordprocessing 206–207, 541–545
Words
in FORTH 1482–1484
Work area, of text-editor 853
Workspace, in memory 210–214

X

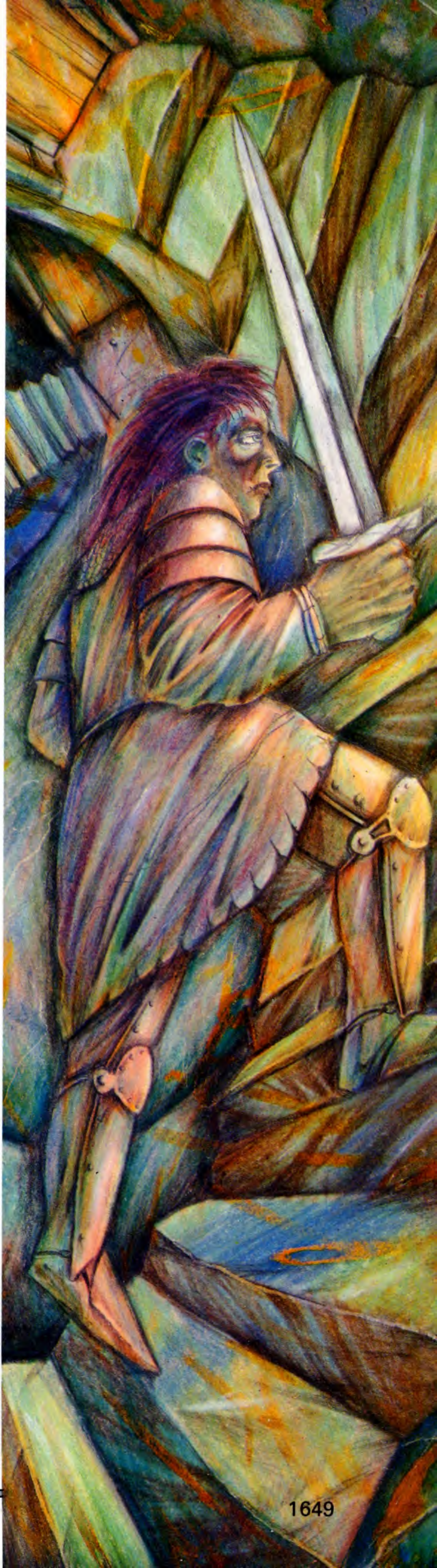
XOR, in FORTH 1533

Y

Yacht dice-throwing game 1589–1595

Z

Zapping sound effect
Spectrum 732
Zero page *Acorn* 215, 582
Zoom effect, in magnification program 1081–1087
ZX printer *Spectrum* 1365–1366



The purpose of INPUT is to teach people to program. Both known and unknown errors ('bugs') do indeed occur in INPUT, as in all other computer programs; we hope the more detailed explanations of programs given in INPUT have made it easier for you. If you are to learn to program remember that professional programmers spend 70 per cent of their time finding and removing bugs.

This is a short course in how to find and kill bugs, though the last section is devoted to advice for beginners and frustration-reducing tips. INPUT itself deals with bugs in an article in Issue 11. Bugs that have to be dealt with from INPUT include both your typing errors and genuine program errors; if you ever want to write programs seriously you will have to learn how to cure bugs without the aid of a guaranteed-correct script of the program. If your hardware can produce it, always print a listing of the program (so that you can see more than the small-screen window on the program), as the first step in debugging it. You will need a printer to write programs to a commercial standard (though a ZX-printer will do as you will be the only person who reads the listings).

Proofreading listings. As we shall see, this is not always the best way of finding bugs, but if done at all it should be done properly. It needs two people (a) because they can keep their place in two listings at once, and (b) because everybody has 'blind spots' and another person will see things differently from you.

Bug-hunting proper: There are at least two different styles, which can be named the 'Private Eye' and the other the 'Interpol' approach. Also given will be some desperation 'Military' methods which can be used as a last resort. The real Private Eye works by selecting and assembling his suspects in a small room and relentlessly spotlighting their defects.

How do you do this? After SAVEing the program you wish to debug, you should RUN it. (Never mind if you haven't got proper data for it; you will be lucky if it runs long enough to need any.) Almost immediately, with luck, it will fall over. Now you have your 'point of failure' – the equivalent of the Private Eye's small room and suspects.

PRIVATE EYE: LESSON 1

What to do with a point of failure. You probably have a computer-generated rude message on the screen. If you don't know what your manual says about this message look it up now before you lose it. (If you haven't got a manual with error messages explained, prepare yourself to program by either buying one or getting one out of your public library.) Jot down on paper any numbers in the message. Now list your version of the line that failed (just that line, listed on the screen). Can you see what is wrong with it? If you have master listing(s) compare it with them. If you can see what is wrong, put it right and score one dead bug. If the line reads something like:

```
8216 /256 SYNTAX ERROR
```

it was that calculation you forgot to put PRINT in front of. Erase it and check if there is a real line 8216 to take its place. Maybe you can't see it, eg:

```
230 IF TAND63=31 THEN T=T-64*INT
(T/64+5)-2)
```

so here's how to find it. First establish if it is there. Type spaces on top of the line number, or delete it, or copy the line without the number and execute it as a direct instruction from the keyboard. It will either give the same message or it won't. Suppose it does. Then the problem is in that line. Simplify the line and retry. If it's an IF line, replace everything after THEN with PRINT "YES"; if it still fails then your problems are before THEN. If you have a complicated expression, erase everything else and put PRINT in front of it. If it still fails unpack it. Keep PRINTing the bits. Frequently it is a READ statement that fails. Look at the DATA that goes with that READ. Check for dots that become commas or vice versa (in PRINT statements check for semicolons that become colons). If the READ is in a loop check the current value of the loop variable. If it is on the first time round, do you have extra DATA before that DATA of the loop? If the last, do you have missing DATA before DATA of the loop?

What to do if the line it fails on ISN'T wrong. That line was written for a purpose and under some circumstances presumably the computer ought to be executing it. Use lesson 2 below to analyse the program to find whether this is one of the times it ought to be executing it. If it isn't you know that it got there at the wrong time and you now know where it ought to have been. Look for the connection. If it ought to have been executing it and it was executing it and it still didn't work, and if it is not wrong, some other part of the program has fed it incorrect data. What are the values of the variables it was working on? (If you need a repeatable way of testing programs that use random numbers, see your manual on how to set the seed for randomizing).

PRIVATE EYE: LESSON 2

How to find a point of failure. Soon your program will stop falling over. (If it loops, BREAK into it, the loop is your point of failure). Now you have more subtle problems associated with "What I was trying to do was...". If it always goes wrong just after you've typed in FRED, hit BREAK just after you've typed in FRED. This gives you a guide to the relevant part of the program. Find out your program structure. With a soft pencil draw fine lines across the page under unconditional GOTOs and heavy ones under RETURNs. Draw braces beside each FOR...NEXT loop. Draw little arrows showing where GOTOs go to, and underline GOSUBs. Where does the line numbering skip to the next multiple of 1? How does it get to the various subroutines? Is there an ON...GOTO for the options of the main menu? What is kept in the biggest array(s)? (Try looking at what puts information in them.) If you understand the structure and roughly where it is going wrong, you have an area of failure. Put extra PRINT statements in the program to identify (a) where it went and (b) what values the variables had. If it shouldn't have been doing the line that it failed in, these print statements will locate the unwanted jump. If some message is wrong, find the PRINT statement that prints it.

PRIVATE EYE: LESSON 3

The magic method. Soon your program will only go wrong some of the time. Now ask the magic question "What (exactly) is the difference between the time(s) that it worked and the time(s) that it didn't? What did I do differently?" Check it, then check which piece of the program could possibly care if you did that.

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

A similar tactic, if the program fails in the middle of a set of (nearly) identical lines, is to ask what is different about the one that failed.

INTERPOL: LESSON 4

Interpol are good at finding suspects given general evidence of a crime. They interrogate everybody (proofreading), and they look at those who are different. Is the fact that this line number isn't a multiple of 10 evidence that it was a late corection (bugs live in clusters)? The longest line in the program is the one most likely to be wrong. Count the items in each DATA statement – why does one differ? It may be legal on the Spectrum to jump to a line that doesn'r exist but it is probably wrong all the same. Suspect unround constants – x/2256 is probably a misprint for x/256. Why does it GOSUB 1300 here when there are lots of GOSUB 1300's? A variable O is probably meant to be Ø. If a variable name only occurs ONCE in the listing of a program it must be a bug.

INTERPOL: LESSON 5

Motive Opportunity Means. If the word SCISSORS appears in a printed message, it must have been produced by a PRINT statement with that in its text, or come from a DATA statement SCISSORS. Which lines had the means to create the hashup I can see? If the loudspeaker howls, which lines look after program sounds that had the opportunity? Which of those lines CAN call the sound routine under any circumstances?

INTERPOL: LESSON 6

When a murder is committed Interpol interrogate the last person to see the victim alive and the first person to see him dead. So, what was the FIRST thing your program did wrong when you ran it? What was the LAST thing your program did before it collapsed? Where did it go wrong?

MILITARY METHODS: LESSON 7

If you have tied the error to a smallish part of the program but just CAN'T see it, pretend to be a computer and carry out each Basic line. This is called hand-tracing. For an assembler program, buy a monitor and 'single-step' through the doubtful area.

MILITARY METHODS: LESSON 8

Try deleting the doubtful bit of program altogether. This can result in the program working, failing the same way (in which case the bit you deleted was innocent), or telling you what is wrong. Reinstate the deleted bit gradually to find where it is wrong, if it is.

MILITARY METHODS: LESSON 9

Delete the doubtful bit as in Lesson 8 and write your own version. You may suddenly understand. Alternatively your new version may work. Alternatively the bug may not go away. Whatever happens you learn something. **General Information:** The longer programs in INPUT have more bugs in them. (Time taken to write a program is proportional to the square of the number of lines in it. So is debugging effort.)

The text compressor, the assemblers and the Hobbies File databases are clean. The Input Hi-res (Commodore) and 'Escape' are least so. Apart from the Hobbies File, most programs in INPUT presented for four machines are four versions of the same program. Try comparing the version for your machine with the versions for the other machines. This often explains obscure methods as well as revealing misprints.

Beginners must: Read the manual that came with the machine. The Commodore inverse-print graphics are in Chapter 5, p.57 and Chapter 4, p.43. The Spectrum manual has how to type keywords in the Appendix. AT AND OR and TO are all keywords, and must be done this way. The BBC-B does not regard upper and lower case as indistinguishable. The Dragon and the BBC, unlike the C-64, may require a space between words if it could be ambiguous. The Plus/4 is NOT a C-64.

Tips to save your strength for debugging and avoid frustration in other directions: When typing in a long program, SAVE once an hour onto tape or disk. VERIFY the last SAVE of each session (BBC-B equivalent is *CAT). Do not SAVE the latest version on top of the last version you SAVEd in case the SAVE itself goes wrong. The BBC is most intolerant of wrong level on the tape recorder, the Dragon of wrong positioning along the tape (you are allowed about 1/10 second of latitude).

Switch the computer off before you unplug peripherals (especially cartridges). About 80 per cent of all Commodore repairs are for people who didn't! Do not type all your accounts into a program until you know it works and will save and reload data reliably. Do not attempt to RUN a program (especially one that POKEs itself or uses maching code) after typing it in before you have SAVEd it. If it goes wrong you may lose the lot. Save assembler source before you try to assemble it.

Vol 1 Nos 1–13 Pages 1–412
Vol 2 Nos 14–26 Pages 413–824
Vol 3 Nos 27–39 Pages 825–1236
Vol 4 Nos 40–52 Pages 1237–1620

Spectrum

Volume 1

Page 12, col. 2, Line 150, add □ between " "

Page 69, col. 3, delete text: Remember to delete it later.

Page 134, col. 3, Line 30, change PAUSE Ø to PAUSE 15

Page 139, col. 2, Line 25ØØ, change LEN to (LEN N$<3)-LEN

3, Line 2520, change LEN to (LEN N$<3)-LEN

Page 201, col. 3, Line 1Ø, change 6 to 7

Page 347, cols. 2 and 3, in Lines 61Ø, 63Ø and 3Ø1Ø IN must be typed; the abbreviated form [I] must not be used.

Page 356, col. 2, Line 5Ø, change <> to ><

Page 380, col. 2, Line 5130, line two, change "add', to "add",

Page 381, col. 3, Line 5350, after GOTO add 9999

Page 382, col. 1, Line 5490, change 542Ø to 532Ø

Page 408, col. 1, Lines 210, 280, change all ˆ symbols to ↑

Volume 2

Page 436, col. 1, Line 70, change 1Ø,21 to 1Ø,Ø21

Page 616, col. 1, delete last sentence under Spectrum editorial

Page 815, col. 3, Line 170, second line, after third symbol add "

Volume 3

Page 940, col. 3, Line 690, add ; after LET b=Ø

Page 984, col. 2, add Line 155 LET h=1

Page 1028, col. 2, Line 130, change "DD" to "CD"

Line 150, add ; after 12 and before NEXT i

Line 160, first line add : after 12 and, second line add : before NEXT i

Line 170, first line, add : after 12 change C$= to C$ add : before NEXT i

col. 3, Line 930, change i,j," " to i,j;"■"

Line 1030, after PRJNT change A to AT

Page 1029, col. 1, Line 1090 reverse P and O should be in graphics mode

Page 1057, col. 1 last group, print should be on separate line at foot

Page 1064, col. 2, Line 1850, change to to TO

Add Line 1930 IF KB=Ø THEN LET X2=7: PRINT#P

Delete Line 1940

Page 1074, col. 1, Line 200, change 2, to 2;

col. 3, Line 660, last line should read GOSUB 56Ø:GOSUB 76Ø: GOTO380

Line 750, change 18Ø to 18,Ø

Line 760, change 16,1) to 16,l;

Page 1075, col. 3, Line 390, change 16,1; to 16,l;

Line 400, last line, change IN7 to INK7

Line 430, add : after 54Ø fourth line, ad : after 6ØØ

Page 1076, col. 1, Line 460, delete one black square

Line 780, delete one white square

Page 1082, col. 1, Line 20, change 85 to 88

col. 2, Line 190, change x>-32768+sp to x<32768+88

Line 200, change x<32768-sp to x>-32768+SP

Line 210, change y>-224ØØ+sp to y<224ØØ-SP

Line 220, change y<224ØØ-sp to y>-224ØØ+SP

Volume 4

Page 1402, col. 2, change call scr to scn

Page 1486, col. 1, Line 530, delete LINE

Line 540 should be 54Ø IF I$=U$ AND TT=1 THEN LET TT=Ø: THEN GOSUB 3Ø4Ø:GOTO 27Ø

Line 640 should be 64Ø LET I=1

Page 1488, col. 1, Line 4510, change Y$+ to Y$=

Line 5000, delete INSTR ROUTINE

Lines 5010, 5020, 5030, 5040 delete and substitute: 5Ø1Ø LET IN=Ø:IF LENY$<=LENX$ THEN RETURN 5Ø2Ø LET Z=(LENX$-LENY$+1) 5Ø3Ø IF Y$= " " THEN LEN IN=Z: LET Z=(LENX$-LENY$-1)

Page 1493, col. 1, Line 2830, delete LINE

col. 2, Line 2930, delete LINE

col. 3, Line 3080, delete LINE

Page 1494, col. 3, Line 3240, delete LINE

Commodore

Volume 1

Page 48, col. 2, Line 70. The symbols are produced by the Shift and Asterisk keys

Page 49, col. 2, Line 1210, the symbols are produced by Cursor/Left keys

Page 114, col. 2, Line 30, change ; to :

Page 128, col. 1, Line 20, change 4,4: to 4,4,7:

Page 143, col. 1, Line 400 should be 4ØØ VV=INT(VAL(VV$)*1ØØ+.5)/1ØØ: VV$=STR$(VV):IFVV=INT(VV) THENVV$=VV$+".ØØ"

Page 196, col. 1, Line 1015, change LE to LE □

Volume 2

Page 429, col. 1, Line 10, delete 3 from end of line

Volume 3

Page 856, col. 2, Line 100, delete B=F at end and add 9Ø

Page 968, col. 1, change LDY#$18 to LDY#$1C

Page 994, col. 1, Line 1160, add ,Ø at end

Page 1019, col. 3, Line 1280, add : after 6)

Page 1031, col. 1, Line 235, change 1Ø" at end to 2Ø"

Page 1066, col. 1, add Line 1755 CD(2)=CD(2)-35

col. 2, Line 2170, before reverse R add reverse heart

Page 1077, col. 3, in text, change 5205, 6025 and 7025 to 52ØØ, 6Ø2Ø and 7Ø2Ø

Page 1098, col. 2, under SPEEDING UP THE PROGRAM, add "Commodore owners should POKE 44,12 before LOADING this program if they intend to RUN it."

Page 1157, col. 1, Line 2500, after DIM add R

Page 1174, col. 2, the C64 symbol should be blue, not black

Acorn

Volume 1

Page 52, col. 3, Line 4, delete + and insert &

Page 125, col. 3, the top paragraph is for the BBC only, not Dragon/Tandy

Volume 2

Page 487, col. 3, Line 940, delete the four ↑ and insert four ˆ

Page 633, col. 2, Line 20. When using the coder/decoder program from parts 20 to 22 to produce the DECODE file for Escape, it is important to change the following two lines: Line 20 should be 20 HIMEM=&79ØØ

Page 653, col. 2, Line 1900 should be 19ØØ *SAVE DECODE 79ØØ 7A8Ø

In addition the coding section from Part 20 *must* be typed in. Users of Basic I must add the following line to the game program from Parts 44 to 48 5000 DEFFNINSTR(A$,B$):IFLENA$ LENB$:=Ø ELSE:=INSTR(A$,B$) and change INSTR in Lines 550,660, 1990 and 3320 to FNINSTR

Volume 3

Page 910, col. 2, Line 920 should be 92Ø DATA 165,12Ø,8

Page 1124, col. 1, Line 120, change 144 to 145

col. 3, Line 540, change TAB to f5

Dragon/Tandy

Volume 1

Page 125, col. 3, the top paragraph is for the BBC not the Dragon/Tandy

Page 215, artwork, 46Ø8 should be 46ØB

Page 355, col. 2, editorial. Delete' Note, on the Tandy you'll have to change the 329 to 282 in Lines 20 and 999'; insert 'Note: Tandy TRS-80 users should delete POKE 329,Ø: from Lines 9Ø and 999.'

A LAST CHANCE TO GET INPUT BINDERS & BACK NUMBERS

A MARSHALL CAVENDISH 38 COMPUTER COURSE IN WEEKLY PARTS
INPUT
LEARN PROGRAMMING - FOR FUN AND THE FUTURE

This is the final issue of *INPUT*—we hope that you have found the series both entertaining and practical!

INPUT BINDERS Have you remembered to order your complete set of binders to display and preserve your collection? The binders are specially designed to enable you to have easy access to all the information in *INPUT* — they are very user-friendly, and look really up-to-date with their smart black finish stamped with eye-catching silver graphics. There are four binders in the complete set — each one holds 13 issues. If you haven't completed your collection of binders, please use the order form supplied on the outside of this issue.

BACK NUMBERS If you have missed any of the 52 parts of *INPUT*, you can order them on the same order form—remember, this is your last chance to complete your collection of back numbers and binders. Thank you for collecting *INPUT* — we've really enjoyed creating it for you!